Sécurité des ressources collaboratives dans les réseaux sociaux d'entreprise

Ahmed Bouchami

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Sécurité des ressources collaboratives dans les réseaux sociaux d'entreprise

THÈSE

présentée et soutenue publiquement le – 02 septembre 2016

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Ahmed BOUCHAMI

Composition du jury

Rapporteurs :	Maryline LAURENT	Professeur, Telecom SudParis
	Abdelmadjid BOUABDALLAH	Professeur, Université de Technologie de Compiègne
Examinateurs :	Salima BENBERNOU	Professeur, Université Paris Descartes
	Yves LE TRAON	Professeur, Université du Luxembourg
	Olivier FESTOR	Directeur de recherche, INRIA
Invités :	Ehtesham ZAHOOR	Maitre de conférence, National University of Computer and Emerging Sciences, FAST-NU, Islamabad, Pakistan
Directeur de thèse :	Olivier PERRIN	Professeur, Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Je tiens à remercier du fond du cœur toutes ces personnes qui m'ont soutenu et aidé pour la réalisation de cette thèse.

En premier lieu, je tiens à remercier le professeur M.Olivier PERRIN pour m'avoir encadré et accompagné tout au long du projet de thèse : un encadrement marqué par la rigueur, la subtilité et le respect. Je remercie également mon ami Ehtesham ZAHOOR de m'avoir soutenu dans des moments difficiles de ce projet et de m'avoir aidé pour la réalisation d'une partie importante de celui-ci. Par ailleurs, je remercie tous les membres de mon jury de thèse qui m'ont honoré d'avoir accepté d'évaluer mon travail et pour l'avoir apprécié.

Je remercie tous les membres de l'équipe COAST que j'ai eu la chance de connaitre pendant mon passage au LORIA, avec une pensée particulière pour Claude GODART pour ces conseils avisés, Élio GOETELMAN et Amina AHMED-NACER pour leur aide précieuse et Jordi MARTORI pour la bonne ambiance au sein de notre bureau.

Je tiens également à remercier les collègues du laboratoire LORIA-INRIA GrandEST pour les agréables moments que j'ai passés en leur compagnie : merci Cheddy, Wazen, Hamma, Mohamed, Mehdi, Rafik, Karim, Yassine, Younes, Slouma, Nabil, Nihel, Mohcen, Tarik, Isabelle ainsi qu'à tous ceux qui ont participé à cette belle aventure. Je remercie du fond du coeur mes amies Manelle, Asma, Khaoula, Ilef et mon camarade Faycel pour les moments de joie qu'on a passés ensemble.

Mes pensées les plus profondes sont évidemment réservées à mes chers parents, et mon adorable femme pour leur soutien psychologique qui m'a inspiré durant ces dernières années.

À ma très chère famille.

Sommaire

Abstract

Enterprise social networks (ESN) have revolutionized collaboration between professional organizations. By means of an ESN, conventional mobility constraints, complex procedures for services exchange and the lack of flexibility and communication are no longer concerns. In this thesis we have worked on the project *OpenPaaS* ESN. Mainly we focused on the management of the access control, which led us to other needs, namely the management of digital identities and their monitoring. We worked primarily on managing the authentication of digital identities within collaborative communities made of heterogeneous enterprises regarding authentication management systems. For this, we have proposed an interoperable architecture for managing federated authentication, allowing thus each enterprise to preserve its (own) authentication mechanism and each principal to perform a single sign on authentication regarding different enterprises. Further, we focused on the management of digital identities accreditations, *i.e.* Access Control. On this aspect, we have proposed a flexible access control model based on a set of identity attributes. We developed this model on the basis of a formal language based on temporal logic, namely the *Event-Calculus* logic. We were thus able to make the sharing of resources fluid and agile, and also able to handle temporary authorizations, *i.e.* delegations. The fluidity and agility of the shares is due to the user-centric resources' sharing in a straightforward manner. In addition, the logical formalism has allowed us to automatically check the access control policies consistency. For enterprises, our access control system gives them the ability to control the user-centric sharing policies through policies based on a risk management mechanism, which make our access control mechanism dynamic. The risk mechanism is based on the *NIST*'s risk definition with an alignment with a set of parameters that include access control in the ESN context. More precisely, the dynamic risk management includes, the collaborative resource's importance, the authentication system's vulnerabilities and trust level reflected through the behavior of each collaborative actor. On this latter aspect of trust, we made an evaluation of trust through the computation of reputation scores based on the history of collaborative interactions of each subject of collaboration. Finally, we have implemented all those security modules and integrate them as a prototype into *OpenPaaS* ESN.

Keywords: Security, Identity Federation, ABAC, access control, delegation, risk management, digital trust.

Résumé

Les réseaux sociaux d'entreprise (RSE) ont révolutionné la collaboration entre les organisations professionnelles. Grâce aux RSEs, les contraintes classiques de mobilité, de procédures compliquées d'échange de services et de manque de flexibilité et de communication en matière de cercles collaboratifs ne sont plus d'actualité. Dans cette thèse, nous avons travaillé sur le projet *OpenPaaS* RSE. Principalement nous nous sommes focalisés sur la partie gestion du contrôle d'accès, qui nous a conduit vers d'autres besoins, à savoir la gestion des identités numériques et leurs supervisions. Nous avons travaillé en premier lieu sur la gestion de l'authentification des identités numériques au sein de communautés de collaboration regroupant des entreprises hétérogènes en matière de gestion de l'authentification. Pour cela, nous avons proposé une architecture de fédération interopérable en matière de gestion de l'authentification, permettant ainsi à chaque entreprise de préserver son mécanisme d'authentification (propre) et aux acteurs de procéder à une authentification unique. Nous nous sommes ensuite concentrés sur la gestion des accréditations des identités numériques (*i.e.* contrôle d'accès). Sur cet aspect, nous avons proposé un mécanisme flexible de contrôle d'accès basé sur un ensemble d'attributs identitaires, que nous avons conçu sur la base d'un langage formel fondé sur la logique temporelle *Event-Calculus*. Nous sommes ainsi en mesure de rendre le partage de ressources fluide et agile, et par ailleurs capables de gérer des autorisations temporaires (*i.e.* les délégations). La fluidité et l'agilité du partage sont dues au fait que nous avons modélisé notre mécanisme de contrôle d'accès de telle sorte que le partage soit basé principalement sur les acteurs de collaboration (*i.e.* user-centric), et ce de la manière la plus simple possible. En outre, le formalisme logique nous a permis de vérifier automatiquement la cohérence des politiques notamment celles liées au partage de ressources. Notre système de contrôle d'accès donne aux entreprises le pouvoir de contrôler de manière abstraite les politiques de partage de ressources définies à l'échelle des acteurs, et ce grâce à des politiques fondées sur un mécanisme de gestion du risque qui émane des requêtes externes de demande d'accès. Les politiques basées sur le risque sont combinées avec les politiques de partage. Dans notre mécanisme de gestion du risque, nous nous sommes basés sur les standards liés au risque (définis par le *NIST*) que nous avons alignés avec des paramètres pertinents pour le contrôle d'accès dans le contexte RSE. Notre gestion dynamique du risque inclut en effet les paramètres suivants : l'importante de chaque ressource collaborative, les vulnérabilités des systèmes d'authentification utilisés pour authentifier les acteurs au sein d'une communauté et la confiance reflétée à travers le comportement de chaque acteur de collaboration. Sur ce dernier aspect de confiance, nous avons procédé à une évaluation de la confiance numérique à travers le cumul de réputations basé sur l'historique d'interactions collaboratives de chaque sujet. Enfin, nous avons développé ces différents modules de sécurité orientés pour le contrôle d'accès dans les environnements collaboratifs socioprofessionnels, et nous les avons intégrés au prototype du RSE *OpenPaaS*.

Mots-clés: Sécurité, Fédération d'identité, ABAC, Contrôle d'accès, Délégation, Gestion du risque, Confiance numérique.

Chapitre 1

Introduction générale

Sommaire

Dans cette partie nous allons en premier lieu introduire le contexte général des travaux réalisés dans le cadre de cette thèse. Ensuite, nous détaillerons les problématiques de nos travaux de recherche menés tout au long de cette thèse.

1.1 Introduction

Cette thèse porte sur la *sécurité* dans les *environnements collaboratifs*. Nous nous sommes particulièrement focalisés dans nos travaux sur la gestion des identités numériques, la gestion des autorisations et la supervision. Les environnements collaboratifs constituent le cadre général qui a influencé notre analyse et par conséquent nos solutions. C'est pourquoi nous commençons par aborder la notion d'environnement collaboratifs afin de mieux élucider le contexte sur lequel sont fondées notre problématique de recherche et nos différentes contributions.

Un environnement collaboratif est principalement un contexte multi-acteurs dans lequel diverses entités collaborent pour la réalisation d'objectifs communs et convergents. L'idée principale d'une collaboration se base sur le partage de tâches et l'échange d'informations et de ressources entre différents acteurs. Ces acteurs peuvent être issus de multiples domaines de compétences, ce qui nous permet de qualifier d'environnement *hétérogène* un environnement collaboratif.

L'évolution des environnements IT a permis de faciliter la collaboration, en banalisant les contraintes de mobilité et de communication. Cela a permis d'élargir le périmètre de collaboration afin de maximiser le rendement et l'efficacité du travail collaboratif. Cependant, un travail de collaboration de bonne qualité exige une bonne gestion au sein de l'environnement collaboratif. L'un des piliers de cette bonne gestion réside dans la capacité et les compétences à s'adapter aux changements permanents des entités collaboratives. Une entité collaborative peut être active (*i.e.* un acteur humain ou un logiciel) ou bien passive (*i.e.* une ressource). Les réseaux sociaux sont l'une des évolutions les plus récentes et les plus riches des environnements collaboratifs. Néanmoins, le challenge d'une bonne gestion au sein d'un environnement collaboratif est accentué quand on rajoute à ce dernier le caractère social.

Une variante sociale d'un environnement collaboratif de travail est appelé *Réseau social d'entreprise* ou *"RSE"*. Un RSE est fondé sur les échanges au sein d'environnements collaboratifs dans un cadre professionnel. La dernière décennie a connu une émergence assez importante des plates-formes dédiées à cette nouvelle dimension des réseaux sociaux, et de nombreux réseaux sociaux d'entreprises ont vu le jour. Parmi les plus connus, on trouve : *Yammer* [1], *Socialtext* [2], *Tibbr* [3], *eXo* [4], *OpenPaaS* [5], *Noodle* [6], *etc*. Une revue complète se trouve dans [48].

Cette thèse a été réalisée dans le cadre du projet industriel *OpenPaaS* en collaboration avec la société *Linagora*. Éditeur de logiciels spécialisé dans l'*Open source*, *Linagora* a intégré le marché des réseaux sociaux d'entreprise avec sa plate-forme collaborative *"OpenPaaS"* et la définit comme suit :"*OpenPaaS est une plate-forme complète permettant de construire et de déployer des applications en s'appuyant sur des technologies éprouvées : messagerie collaborative, intégration (Enterprise Service Bus) et Business Process Management (BPM). OpenPaaS est destinée aux applications collaboratives d'entreprises déployées dans des Clouds hybrides (privé/ publique)*". En d'autres termes, *OpenPaaS* est une plate-forme collaborative de type *PaaS* [7] [120] qui permet à plusieurs entreprises de collaborer dans le cadre d'un RSE.

Comme le montre la figure 1.1, une autre particularité des RSEs en général et *OpenPaaS* en particulier concerne la dimension ubiquitaire de tels environnements [158, 135], dans la mesure où plusieurs périphériques peuvent être utilisés en tant que point d'accès au RSE. À titre d'exemple, nous avons les smartphones, les tablettes, les ordinateurs portables, les assistants numériques per-

1. https://www.yammer.com/
2. http://www.socialtext.com/
3. http://www.tibbr.com
4. http://www.exoplatform.com/company/en/resource-center
5. https://research.linagora.com/display/openpaas/Open+PaaS+Overview
6. https://vialect.com/
7. Platform-as-a-Service

sonnels (*Pocket PC*), les ordinateurs de bureau, les serveurs, les services web, *etc*. Par ailleurs, et d'un point de vue plus technique, *OpenPaaS* permet à plusieurs partenaires d'héberger des ressources, des services, ainsi que des processus dans une PaaS collaborative.

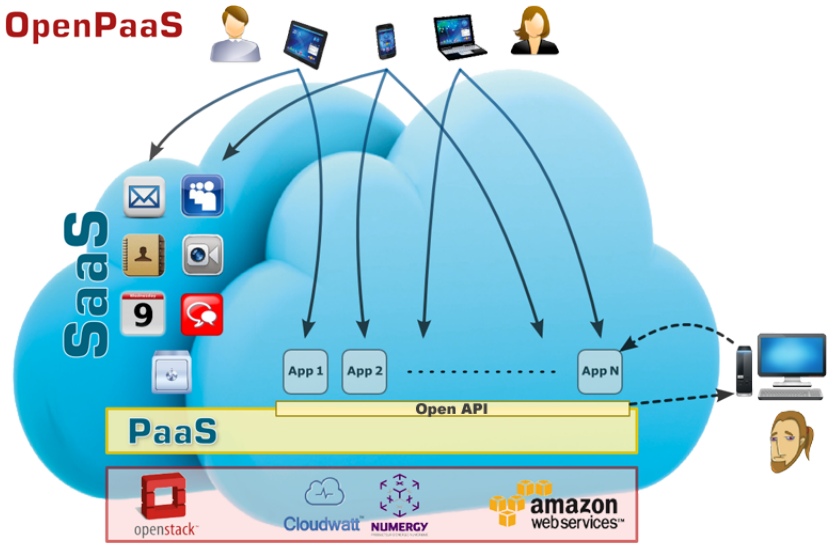

FIGURE 1.1 – Aperçu global sur la conception d'*OpenPaaS* et ses différents services [1].

Dans la section suivante, nous présentons le contexte général dans lequel ont été réalisés les travaux de cette thèse.

1.2 Contexte général

Depuis la dernière décennie, les plates-formes collaboratives socioprofessionnelles ont changé le comportement des entreprises en matière de gestion de la collaboration interentreprises. Ce changement s'est montré à travers une facilité de communication, due avant tout aux nouvelles technologies de gestion des ressources et de l'information, en particulier le *Cloud Computing*. En effet, en plus d'avoir porté les technologies de l'information à de très grandes échelles, le *Cloud-Computing* a permis de promouvoir de nouvelles manières de collaborer à travers la communication et le partage de services, de ressources et d'informations. Il est désormais possible de partager des données entre des applications déployées dans le Cloud et des bases de connaissances d'entreprises [1].

Par exemple, on peut citer le partage de ressources au moyen de parties tierces à travers des plate-formes dédiées comme : *GitHub* [8], *Dropbox* [9] ou *Agora* [10]. Par ailleurs, nous avons également le microblogging qui consiste à échanger des informations en instantané et à grande échelle à travers des cercles (*Google+*) ou des *#Hashtag* (*Twitter*). La principale motivation derrière ces innovations est de simplifier davantage tout obstacle technologique susceptible de ralentir le processus d'échange

8. http://www.github.com
9. http://www.dropbox.com
10. http://www.agora-project.net

collaboratif de ressources entre des entités hétérogènes et distribuées.

Un RSE est principalement composé d'organisations professionnelles. Une organisation est l'abstraction de tout type d'entreprise fondée sur la collaboration entre multiples membres pour des objectifs communs. Une organisation peut également faire référence à un domaine réunissant plusieurs entités interagissant dans un cadre coopératif, *e.g.* le domaine universitaire. Pour lever toute ambiguïté de langage, nous généralisons l'utilisation du terme *"entreprise"* pour désigner tout type d'organisation et de domaine.

Dans un environnement collaboratif restreint, une confiance mutuelle peut être établie entre les partenaires à propos de l'échange d'informations, les conduisant à la formation d'un cercle de confiance commun. Ce genre de cercle de confiance est référencé dans la littérature par le terme fédération [181]. Ainsi, nous considérons *OpenPaaS* comme étant un environnement de fédération. En effet, l'aspect collaboratif d'*OpenPaaS* repose principalement sur l'idée de créer des ***Communautés*** virtuelles pour rassembler différentes entreprises. Ceci vise à faciliter les interactions entre différents acteurs ayant des compétences complémentaires en les réunissant autours de projets collaboratifs communs.

Pour résumer, nous considérons *OpenPaaS* RSE comme étant un environnement *fédéré, distribué, dynamique* et *hétérogène*. Ces propriétés sont illustrées sur la figure 1.2. Cette figure décrit le cadre général des communautés dans le RSE *OpenPaaS* dans lequel plusieurs entreprises collaborent au sein de la même communauté. De nouvelles entreprises, avec de nouveaux acteurs et ressources, peuvent à tout moment rejoindre la communauté de collaboration. Au niveau de chaque entreprise, des acteurs gèrent les ressources de l'entreprise pour les échanger avec d'autres acteurs appartenant à des entreprises partenaires au sein de la communauté en question. Ces ressources peuvent changer à tout moment, elle peuvent être partagées avec de nouveaux acteurs, supprimées, modifiées, *etc*. D'un point de vue sécurité, chaque entreprise gère à sa manière les identités et les accréditions d'accès à ses ressources (IAM *"Identity Access Management"*). Cela met en évidence l'hétérogénéité omniprésente au sein de la communauté en matière de gestion de l'intégrité et de la confidentialité des ressources collaboratives.

Dans la section suivante, nous introduisons les concepts fondamentaux liés à la gestion de la sécurité des échanges de ressources et d'informations dans les environnements collaboratifs.

1.3 La sécurité dans un RSE

La sécurité est l'un des aspects majeurs d'une bonne gestion au sein d'un environnement collaboratif socioprofessionnel. Ainsi, dans de tels environnements, à l'image de *OpenPaaS*, les besoins en termes de sécurité restent classiques, à savoir, la gestion des identités (**A***uthentication*), le contrôle d'accès (**A***uthorization*) et la supervision (**A***udit*). Ces services sont référencés dans la littérature sous l'acronyme ***AAA*** [65, 66, 134].

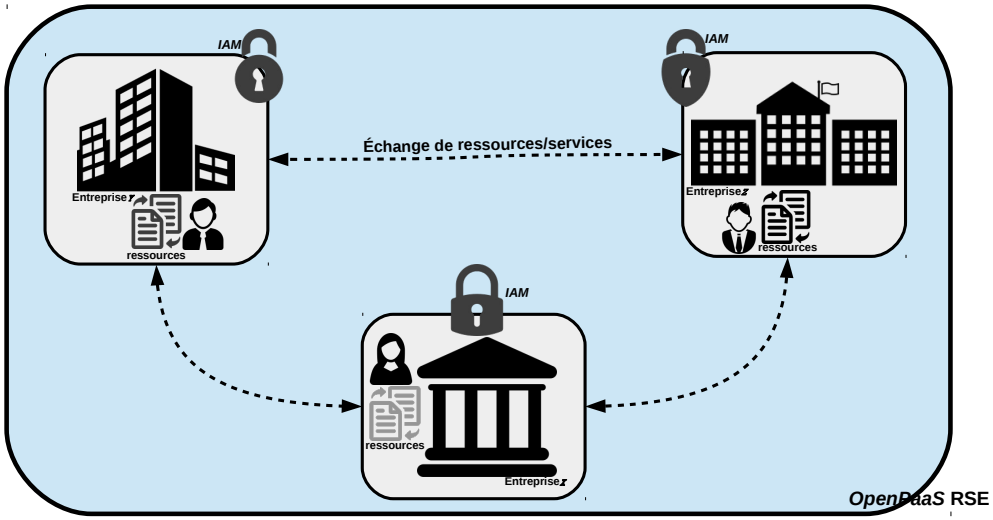

FIGURE 1.2 – Vue globale sur une communauté *OpenPaaS*.

1.3.1 Authentification

L'authentification est souvent le premier *volet* de la protection de ressources vis-à-vis d'entités ayant une identification numérique. L'authentification consiste à vérifier l'authenticité de l'identité d'un utilisateur à travers des *identifiants* que ce dernier fournit au système, *e.g.* login/mot-de-passe. Dans une procédure d'authentification, les identifiants d'un utilisateur sont comparés avec ceux stockés dans le système. Si une identité stockée correspond aux identifiants fournis par l'utilisateur, ce dernier sera alors *authentifié*. Dans le cas contraire, l'acteur ne sera pas authentifié. Techniquement, plusieurs dispositifs peuvent être utilisés dans la procédure d'authentification, à savoir la mémoire de l'utilisateur (Login/mot-de-passe, code pin, *etc.*), les systèmes de codage aléatoire (envoi d'un sms sur le téléphone portable personnel), ou les procédés biométriques (empreinte digitale, reconnaissance vocale/rétinienne, *etc.*).

D'autres mécanismes d'authentification avec des configurations plus avancées existent, comme *OAuth* [110], *OpenID* [162], ou *SAML* [90] [11], dont certains très adaptés aux environnements distribués et fédérés. Ces mécanismes (ou protocoles) se basent sur le principe d'authentification unique. L'authentification unique (SSO) permet à un utilisateur de s'authentifier une seule fois pour accéder à de multiples services distants protégés par le même protocole d'authentification unique utilisé (*i.e.* lors de la première authentification unique de l'utilisateur). Dans un environnement collaboratif, l'authentification unique peut être très intéressante. En effet, nous sommes dans un environnement distribué qui regroupe plusieurs entreprises, chacune protégeant ses ressources d'une manière indépendante.

11. Plus de détails sur ces mécanismes seront présentés dans le chapitre (**cf. État de l'art**).

1.3.2 Autorisation

Bien que l'authentification soit une phase importante pour la protection de ressources numériques, elle reste insuffisante notamment dans un environnement RSE tel que *OpenPaaS*. En effet, l'authentification permet (dans un cas optimal [12]) de vérifier que l'identité de l'acteur qui réclame l'accès derrière une requête est la bonne, mais ne garantit pas que cet acteur accède seulement aux ressources qui lui sont autorisées. Plus précisément, nous sommes dans un environnement multi-acteurs dans lequel sont partagées plusieurs ressources appartenant à différents acteurs (ou entreprises). Par conséquent, nous avons besoin de lier chaque ressource à son propriétaire et éventuellement ses collaborateurs avec qui il la partage, et surtout de s'assurer que seuls les acteurs autorisés y auront accès. Pour résumer, dans un environnement collaboratif social (*i.e.* ouvert), toute ressource partagée est par défaut interdite d'accès à tous les acteurs à l'exception de ceux qui ont bénéficié d'une autorisation d'accès définie par le propriétaire de la ressource en question, et vérifiée par le système. La procédure d'*autorisation* se charge de cette tâche de vérification de droits d'accès. En d'autres termes, la procédure d'autorisation détermine *qui a accès à quoi et comment*, et ce au moyen de permissions (ou autorisation) d'accès prédéfinies impliquant principalement : le *sujet*, l'*objet* et l'*action*.

Une autorisation d'accès est donc une description d'un droit d'accès en faveur d'un acteur donné sur une ressource donnée avec une action précise. Cette description est la plus petite unité de l'ensemble des accords qui régissent les interactions entre différents utilisateurs dans un système collaboratif. Dans ce contexte, un accord est appelé une *politique*, qui est basée sur le regroupement de *règles* de contrôle d'accès. Cependant, une règle n'est pas toujours synonyme d'autorisation, car une règle peut également être définie pour interdire l'accès à un acteur donné. Par conséquent, une politique peut également rejeter une requête de demande d'accès.

Dans une communauté RSE, le principal scénario de consommation (*i.e.* d'utilisation) d'une ressource est déclenché par un acteur au moyen d'une *requête de demande d'accès*, qui sera confrontée par le mécanisme d'autorisation aux politiques de contrôle d'accès enregistrées dans le système. Si les attributs de la requête (*i.e.* la *cible* de la requête) correspondent à ceux d'une règle d'autorisation du système, la requête sera approuvée donnant ainsi le droit d'accès à l'acteur en question, sinon la requête sera rejetée. Cela signifie par ailleurs que, dans le contexte de communauté de collaboration, chaque partage de ressource nécessite la définition d'un droit d'accès précisant l'acteur autorisé (*le sujet*) à y accéder et de quelle manière (*l'action*).

Par ailleurs, les besoins en matière d'autorisation dans un contexte RSE s'étendent à de nouvelles formes de collaboration, comme c'est le cas pour la *délégation*. Une délégation est un transfert temporaire des droits d'accès d'un acteur chargé de certaines tâches à un autre acteur pour les accomplir. Dans cette optique, on distingue la nature éphémère d'un tel transfert de droits qui doit être basé sur la notion de *temps*. En outre, cette forme d'autorisation temporaire est étroitement liée à la confiance. Plus précisément, pour qu'un acteur permette à un autre d'agir pour son compte, une forme de confiance, définie par un lien professionnel ou social, est forcément présente entre les deux

12. Pas de défaillance du système de gestion des identités et de l'authentification.

acteurs. En effet, dans un cadre professionnel, la confiance peut faire référence à plusieurs aspects collaboratifs tels que, les compétences professionnelles, la bonne réputation, la fiabilité, *etc*. L'aspect de confiance est traité sous la sphère du dernier **A** des services **AAA**, à savoir, l'*Audit*.

1.3.3 Audit et gouvernance

La confiance professionnelle est monitorée au moyen de la supervision (ou monitoring) du comportement des acteurs de collaboration. Plus précisément, la supervision est un processus de suivi et d'analyse des traces (ou l'historique) d'interactions de chaque acteur. L'historique d'interaction inclut des fichiers logs, des messages, ou le temps d'utilisation de ressources, et les résultats d'analyse peuvent être exploités dans diverses domaines notamment dans la sécurité, dans les études financières, les études des tendances sociales. Dans cette thèse, nous exploitons l'historique d'interaction d'un acteur à des fins de sécurité.

D'un point de vue sécurisation des échanges collaboratifs de ressources, nous mettons l'accent sur la réputation de chaque acteur afin d'étudier ses différents changements comportementaux, et ce d'une manière dynamique et continue. En effet, le comportement d'un acteur est imprévisible dans un contexte RSE. Par exemple, un acteur change brusquement d'attitude et procède à de nombreuses tentatives d'accès non-autorisées sur des ressources déployées au sein de sa communauté. En outre, la supervision peut être favorable aux entreprises afin de réviser des autorisations d'accès trop statiques ou éventuellement obsolètes.

L'exploitation de la supervision du comportement des acteurs au profit des entreprises peut être très intéressante, car chaque entreprise sera ainsi en mesure d'adapter ses politiques de contrôle d'accès par rapport aux évolutions des réputations des sujets externes [13]. De la sorte, chaque entreprise peut déléguer la définition des autorisations simples (*i.e.* sujet/objet/action) à ses acteurs et avoir par dessus un contrôle via des politiques plus abstraites. Cela permettra de faciliter les échanges collaboratifs en simplifiant la procédure de partage de ressources. Cela dit, considérer la confiance comme unique paramètre peut s'avérer insuffisant pour l'entreprise afin de contrôler les autorisations de partage de ressources d'une manière à la fois efficace et abstraite.

Imaginons qu'un acteur fasse une erreur dans la définition d'une autorisation d'accès et se trompe par exemple de sujet bénéficiaire. Ainsi, pour l'entreprise (propriétaire de la ressource), l'impact d'un accès non-autorisé sur l'intégrité et la confidentialité de la ressource va dépendre de l'importance (sensibilité) de cette dernière. Par conséquent, à l'image de ce dernier paramètre de sensibilité de la ressource, il est possible que d'autres paramètres soient aussi importants. Pour une telle hypothèse, il est nécessaire d'identifier les paramètres les plus pertinents pour le contrôle d'accès dans le contexte des RSEs, et les combiner par la suite. Un mécanisme de *gestion du risque* [151] d'une requête de demande d'accès peut être un bon support pour mettre en œuvre (ensemble) les paramètres recherchés. Le risque peut être considéré comme étant la probabilité d'une menace et son impact sur le système.

13. Appartenant à d'autres entreprises de la communauté dont certaines peuvent être des concurrentes, *i.e.* conflit d'intérêts.

1.3.4 Synthèse

En résumé, nous avons travaillé dans le cadre de cette thèse sur la modélisation et la réalisation d'un système de sécurité destiné aux plate-formes collaboratives sous forme de réseaux sociaux d'entreprise. Ce système vise principalement à contrôler l'accès d'une manière dynamique aux ressources collaboratives vis-à-vis des identités numériques. Par conséquent, notre système est fondé sur un module de gestion interopérable et fédérée des identités numériques, un module de contrôle d'accès flexible, et enfin un module de monitoring qui permet d'évaluer la confiance numérique des entités actives, et ce afin de prendre en compte l'aspect dynamique du module de contrôle d'accès.

Dans la section suivante, nous allons résumer nos solutions proposées par rapport au contexte et objectifs de cette thèse.

1.4 Résumé des contributions de la thèse

Dans cette section, nous donnerons un aperçu sur les différentes contributions proposées dans le cadre de cette thèse. Les problématiques ayant conduit à ces contributions seront détaillées dans le chapitre suivant (**cf. Problématique et motivations**). Nos contributions couvrent les trois aspects : Authentification, Autorisation et Audit. Pour la partie *Authentification*, nous nous sommes basés sur une architecture de collaboration adaptée aux communautés RSEs pour la conception d'une fédération interopérable en matière d'authentification interentreprises. Concernant la partie *Autorisation*, nous avons proposé un mécanisme complet (conception et mise en œuvre) de contrôle d'accès. Enfin, la partie *Audit* inclut un mécanisme de gestion du risque basé principalement sur une évaluation en temps réel de la confiance et la réputation des acteurs collaboratifs.

1.4.1 Conception d'architecture et gestion des identités -Authentification-

La collaboration dans OpenPaaS est fondée sur la notion de communauté qui consiste en un cercle regroupant différents types de ressources ainsi que des acteurs et des entreprises de multiples compétences et disciplines. En effet, OpenPaaS est l'ensemble qui regroupe l'intégralité des communautés de collaboration. En outre, une ressource, un acteur ou une entreprise peuvent faire partie de plusieurs communautés en même temps. Cependant, d'un point de vue sécurité nous considérons que chaque communauté est indépendante en matière de gestion des accréditations des identités des acteurs vis-à-vis des ressources collaboratives. Pour la gestion d'identité, nous avons tiré profit de la confiance qui lie les entreprises dans le cadre collaboratif et professionnel, en l'occurrence la communauté. Cette confiance se traduit par une dérogation (dispense) dans les procédures d'authentification (souvent très strictes) à l'égard des identités d'acteurs qui appartiennent aux autres entreprises de la communauté. Cette notion de confiance interentreprises est appelée *fédération*.

Une gestion fédérée d'identités est synonyme d'une confiance mutuelle qui s'installe entre les entreprises concernant l'identification des acteurs. Par exemple, le service *Eduroam "education roaming"* dans le domaine académique permet à un étudiant de l'*Université de Lorraine* d'avoir accès à internet dans d'autres universités (conventionnées) en s'identifiant au moyen de son identité de l'*Université de*

Lorraine. Cela signifie que les autres universités partenaires de l'*Université de Lorraine* font confiance à cette dernière concernant l'identification de son étudiant. Pour résumer, nous considérons chaque communauté dans *OpenPaaS* comme une fédération. Néanmoins, faute d'incompatibilité technique, les partenaires dans un cadre fédéré sont obligés d'utiliser le même protocole de gestion d'identité [2]. Par exemple, un jeton d'authentification issu du protocole *OpenID* ne peut pas être vérifié par un autre protocole d'authentification, comme *OAuth* [14]. Cependant, dans notre contexte, les communautés sont hétérogènes, ce qui signifie qu'elles n'utilisent pas forcément les mêmes protocoles pour la gestion de l'authentification. Ainsi, notre première contribution s'est focalisée sur la gestion de cette hétérogénéité en proposant un mécanisme d'authentification interopérable qui permettra à chaque entreprise de garder ses préférences vis-à-vis du protocole d'authentification utilisé pour la gestion des identités numériques collaboratives de ses acteurs. Pour cela, nous nous sommes basés sur un outil appelé *"LemonLDAP"*, que nous avons adapté à notre architecture afin qu'il puisse couvrir cette hétérogénéité. Cette contribution a été publiée dans la conférence internationale *I-ESA 14* [39].

Par ailleurs, en matière de gestion des identités et des accréditations, notre conception des fédérations tient compte des trois aspects suivants. Premièrement, la *collaboration interopérable sur le long terme* tient compte de l'hétérogénéité omniprésente au sein de chaque fédération. Cela nécessite d'un côté d'avoir au niveau de chaque fédération un service centralisé de gestion des ressources et des autorisations, et requière d'un autre côté un *faible couplage* entre les entreprises et le RSE. Enfin, la collaboration *temporaire de courte durée* met l'accent sur la facilité dans la gestion (intégration temporaire et/ou suppression) de nouveaux acteurs collaboratifs dans une communauté.

Dans le contexte *OpenPaaS*, nous considérons ces trois propriétés comme étant des besoins importants sur le plan architectural. Ils nous permettent de faire face à plusieurs enjeux, tels que le passage à l'échelle (fluidité d'échanges collaboratifs), l'interopérabilité et la sécurisation de données.

Pour répondre à ces besoins, nous nous sommes basés sur une conception *hybride* des fédérations, *i.e.* un compromis entre une configuration centralisée et une configuration décentralisée d'architecture de collaboration. L'idée est de centraliser la gestion des identités et des politiques et de décentraliser l'hébergement des ressources physiques ainsi que la gestion des règles et politiques d'autorisation d'accès qui les régissent au niveau des entreprises propriétaires et leurs acteurs. Ainsi, le déploiement des ressources se fait au moyen d'une procédure de virtualisation et l'accès sera au niveau des serveurs de l'entreprise propriétaire. La gestion centralisée des identités a pour but de rendre le contrôle d'accès interopérable vis-à-vis des différentes entreprises faisant partie d'une même fédération de collaboration. Cela permet d'homogénéiser la sémantique des attributs qui sont éventuellement initialement différents au niveau de chaque entreprise, et ce en accord avec le contexte de la communauté en question. Ainsi, les attributs d'un utilisateur appartenant à une entreprise donnée peuvent être facilement confrontés aux politiques des autres entreprises partenaires. Quant à la virtualisation et la décentralisation des ressources collaboratives, elles permettent à chaque entreprise de garder le contrôle sur l'intégrité et la confidentialité de ses ressources collaboratives.

14. Cet aspect sera davantage détaillé dans le chapitre **cf. Problématique et motivations**

1.4.2 Contrôle d'accès et supervision -Autorisation et Audit-

À la différence des modèles classiques de contrôle d'accès tels que ceux basés sur l'identité unique de l'acteur ou sur son rôle, le notre est basé sur un ensemble d'attributs qui permettent de définir le profil de l'acteur. Ce choix nous a permis d'avoir une meilleure flexibilité pour la définition des règles de contrôle d'accès.

Dans notre modèle de contrôle d'accès, nous avons deux niveaux de politiques, à savoir le niveau acteur (*user-centric*), et le niveau entreprise. La politique de niveau acteur contient des règles définies par un acteur pour chaque partage de ressource collaborative avec un autre acteur collaborateur. La règle définie par un acteur prend une forme simple impliquant le sujet (*i.e.* le collaborateur), la ressource à partager et l'action autorisée. À l'inverse, la politique établie au niveau de l'entreprise est plus générique, *i.e.* abstraite. Ces politiques abstraites visent à contrôler les politiques de partage de ressources définies par des acteurs. Une politique d'entreprise se focalise davantage sur le contexte de collaboration que sur chaque instance de partage de ressource. Par conséquent, pour chaque requête de demande d'accès les politiques des acteurs et des entreprises (impliquant le sujet de la requête en question) sont combinées afin de prendre une décision finale d'approbation ou de rejet d'une demande d'accès. Cette configuration du mécanisme de contrôle d'accès permet d'un côté de simplifier et d'accélérer la collaboration grâce aux politiques acteurs, et d'un autre côté de préserver l'autonomie des entreprises en leur permettant de contrôler les politiques définies par leurs acteurs.

Dans un RSE en général, la plupart des acteurs collaboratifs sont des utilisateurs humains. Un utilisateur humain peut se tromper et par conséquent mal définir une règle de contrôle d'accès. Cela peut être dû à sa non expertise en matière de contrôle d'accès ou tout simplement à un oubli (voire même une faute de frappe). Par conséquent, il est important de vérifier la cohérence de toute règle composant une politique définie au niveau de l'acteur avant de la combiner avec la politique d'entreprise. En effet, une éventuelle incohérence dans les politiques peut mener le système à des états d'insécurité dus à des erreurs de décisions. C'est pourquoi, nous avons conçu notre modèle de contrôle d'accès à base d'un *langage formel* basé sur la logique temporelle du premier ordre *Event-Calculus* avec son raisonneur *Dec-Reasoner* [15] comme moteur d'évaluation et vérification automatique des politiques de contrôle d'accès. En outre, l'aspect temporelle de la logique *Event-Calculus* nous a permis de modéliser les *permissions temporaires*, *i.e.* les *délégations* de droits d'accès aux ressources collaboratives.

Le contexte de l'environnement de collaboration est considéré sous le volet du contrôle d'accès par le biais de l'évaluation du *risque* des requêtes. Ce dernier inclut la menace encourue à travers la requête du sujet, l'impact provoqué dans le cas où la ressource en question est compromise, et la vulnérabilité du mécanisme d'authentification utilisé pour certifier l'identité du demandeur d'accès. Le premier facteur concerne la menace que représente le sujet d'une requête de demande d'accès. Cette menace est mesurée sur la base d'une estimation de la confiance numérique du sujet à travers la **supervision** de son comportement dans sa communauté. Le deuxième facteur se focalise sur l'impact d'un éventuel accès non autorisé sur la ressource demandée. L'impact est mesuré par rapport à

15. *http://decreasoner.sourceforge.net/*

l'importance de la ressource à travers le nombre d'acteurs impliqués dans la consommation et/ou la propriété de la ressource. Le dernier facteur pour l'évaluation du risque est relatif à la vulnérabilité de l'environnement collaboratif. La vulnérabilité consiste en un indice de fiabilité de l'authenticité des identités des sujets issus de leurs entreprises respectives. En d'autres termes, la vulnérabilité d'un environnement est implicitement reliée au mécanisme d'authentification. Par exemple, l'authenticité d'une identité gérée avec un mécanisme *Pin* est moins fiable qu'une validation en deux étapes [16]. Sur la base de cette réflexion et ces paramètres, nous avons proposé notre métrique d'évaluation du risque des requêtes dans la fédération collaborative.

Par ailleurs, notre modèle de contrôle d'accès tient compte du contexte d'une manière plus approfondie, et ce par le biais d'une approche *événementielle*. Nos règles sont définies sur la base du paradigme *ECA : Event-Condition-Action* [91]. Cela nous a permis de modéliser les différents changements dynamiques en temps réel des états des objets au sein du contexte. Ces changements sont dus aux flux continus d'événements à travers le temps. Ces changements sont dans notre modèle directement impliqués dans les décisions de contrôle d'accès comme c'est le cas pour les *délégations*.

Ces différentes contributions ont été publiées dans plusieurs conférences internationales [199, 40, 38].

1.4.3 Mise en œuvre

Le prototype de gestion des identités numériques et leurs autorisations pour la plate-forme collaborative sous forme de réseau social d'entreprise à été validé et mis en œuvre dans le cadre du projet *OpenPaaS*. Les principaux composants logiciels sont mis en œuvre en tant que services web RESTfull pour s'adapter aux plates-formes basées sur des architectures Cloud. Le premier composant, à savoir l'outil de gestion des identités, a été mis en œuvre sur la base de l'outil *LemonLDAP : :NG*.

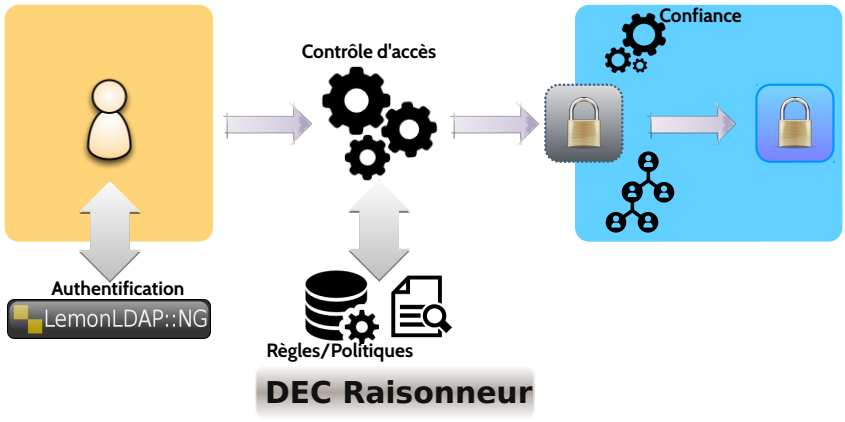

FIGURE 1.3 – Prototype vue globale.

Le second composant concerne la gestion du contrôle d'accès. Ce dernier a été mis en œuvre

16. *http://en.wikipedia.org/wiki/Two−step_verification*

en tant que service web à base du raisonneur *Event-Calculus "Dec-reasoner"*, utilisant la base de données *MongoDB* pour le stockage des règles de contrôle d'accès. Enfin, le dernier composant de supervision et d'évaluation de confiance est aussi mis en œuvre sous forme de service web au niveau de chaque communauté. Ce composant se base sur les informations historiques des utilisateurs d'une communauté pour calculer et mettre à jour les valeurs de confiance numérique de chaque acteur. Ces informations historiques sont stockées sous forme de fichiers logs au niveau de la base de données où sont stockés les attributs des identités des acteurs internes de la communauté.

1.5 Résumé du sujet de la thèse

Cette thèse porte sur la sécurité des échanges collaboratifs au sein d'un réseau social d'entreprise *"RSE"*. Un réseau social d'entreprise est un environnement collaboratif dans lequel le principal objectif est la facilitation d'échanges d'informations et de ressources entre des organisations professionnelles. Par conséquent, l'environnement RSE introduit plusieurs propriétés que nous devons prendre en considération dans la définition de notre problématique de conception d'un système de sécurité d'échanges collaboratifs. Ces particularités du RSE concernent notamment :

- la facilité et l'agilité de partage de ressources,
- l'enjeu très important concernant la confidentialité des ressources entre des entreprises concurrentes (i.e. conflit d'intérêts),
- la dynamicité des propriétés des entités de collaboration (*i.e.* acteurs et ressources),
- l'hétérogénéité des différentes entreprises en matière d'administration,
- la sous traitance des activités nécessitant une continuité d'exécution dans le temps (*i.e.* délégation).

Nos objectifs de sécurité se focalisent principalement sur la gestion des autorisations d'accès. Cette gestion requiert la gestion des identités numériques (qui seront par la suite accréditées), et s'étend par ailleurs sur une supervision des comportements des entités de collaboration au sein des environnements de collaboration. Ces objectifs sont contextualisés dans le projet *OpenPaaS* qui consiste à mettre en œuvre un RSE sur la base d'une plateforme PaaS collaborative. Par conséquent, nous devons intégrer nos solutions de sécurité dans la plateforme *OpenPaaS* sous forme de services web [17].

1.6 Organisation de la thèse

Dans cette thèse, nous mettons en œuvre un mécanisme de sécurité d'échanges collaboratifs dans les environnements collaboratifs de type réseaux sociaux d'entreprises RSEs. Nous avons développé un mécanisme de contrôle d'accès ainsi qu'un mécanisme d'authentification et fédération d'identités numériques favorables aux contextes RSE.

17. APIs.

Outre la conception d'une architecture d'authentification fédérée et interopérable dans un environnement où l'hétérogénéité en matière de mécanismes d'authentification est omniprésente, l'originalité de cette thèse est le développement d'un mécanisme de contrôle d'accès flexible et dynamique basé sur un formalisme logique [18] qui respecte l'autonomie des entreprises [19] et qui est capable de gérer les délégations [20].

Ce mémoire est ainsi organisé.

- Dans le chapitre 1 nous présentons le contexte général dans lequel les travaux de cette thèse ont été réalisés, à savoir les environnements collaboratifs de type réseaux sociaux d'entreprise. Nous commençons d'abord par une introduction sur les environnements collaboratifs en général et les réseaux sociaux d'entreprise en particulier. Ensuite, nous abordons la sécurité dans ces environnements et ce sur trois aspects, à savoir l'authentification, l'autorisation et le monitoring. Puis, nous présentons un résumé des contributions de cette thèse. Enfin, nous présentons un résumé du sujet de cette thèse intitulée *La sécurité des ressources collaboratives dans les réseaux sociaux d'entreprise*.
- Dans le chapitre 2 nous détaillons la problématique de cette thèse. Mais d'abord nous commençons par donner des exemples de motivations inspirés de cas réels de collaboration avec des scénarios qui mettent en évidence certaines failles de sécurité dans le contexte RSE. Concernant la problématique, nous clarifions les challenges de nos objectifs de sécurité (*cf résumé du sujet de la thèse*) par rapport au contexte RSE et ce pour chacun des trois aspects de sécurité (authentification, autorisation et audit) présentés dans le chapitre 1.
- Dans le chapitre 3, nous allons présenter un état de l'art portant sur des travaux ayant été réalisés dans des contextes similaires au notre, à savoir la gestion des identités numériques et des autorisations de contrôle d'accès dans les environnements collaboratifs. Nous présenterons par ailleurs des concepts que nous avons utilisés dans l'élaboration de nos solutions.
- Dans le chapitre 4, nous détaillons nos contributions sur la gestion de l'authentification. En premier lieu, nous présentons les trois types de collaboration dans les environnements collaboratifs distribués qui nous ont inspirés pour la conception de notre architecture de collaboration sur l'aspect de gestion et administration des identités numériques et des ressources collaboratives. En second lieu, nous abordons l'aspect de fédération des identités et la gestion de l'authentification dans un environnement hétérogène.
- Dans le chapitre 5, nous abordons la gestion du contrôle d'accès aux ressources partagées au sein des communautés de collaboration RSE. Cette partie englobe une représentation abstraite et une représentation formelle du modèle de contrôle d'accès. En outre, nous abordons les règles de contrôle d'accès temporaires, *i.e. les délégations*.
- Dans le chapitre 6, nous allons aborder l'évaluation du risque d'une requête de demande d'accès. Nous rappelons brièvement en premier lieu le contexte et la motivation qui nous ont orienté vers une conception d'un modèle de contrôle d'accès dynamique basé sur la gestion du

18. favorable à la vérification automatique des politiques de contrôle d'accès
19. en matière de sécurité de leurs ressources collaboratives
20. permissions temporaires

risque. Ensuite, nous parlons du principe d'alignement des concepts standards de gestion du risque avec ceux du contrôle d'accès et les environnements collaboratifs RSE. Nous finirons avec une étude expérimentale qui illustre l'apport du mécanisme de gestion du risque en matière de décision de contrôle d'accès.

- Par rapport à nos objectifs de sécurité [21], le chapitre 7 s'inscrit dans le cadre du monitoring, *i.e.* la supervision des comportements des identités numérique. Dans ce chapitre nous allons détailler notre approche dynamique d'évaluation de la réputation et la confiance numérique des sujets de collaboration au sein des communautés RSE. En premier lieu, nous allons introduire le contexte avec les définitions permettant d'élucider certaines notions et termes utilisés dans le cadre de ce chapitre. Ensuite, nous présenterons nos algorithmes d'évaluation de la réputation et de la confiance. Avant de conclure, nous illustrons l'efficacité de nos procédures d'évaluation de la réputation et de la confiance à travers une étude expérimentale dans laquelle nous observons l'évolution de la réputation et de la confiance par rapport aux comportements de sujets sur une succession de sessions collaboratives dans une communauté de collaboration.
- Dans le chapitre 8, nous présentons les différentes APIs que nous avons développées et intégrées dans le cadre de la plate-forme OpenPaaS RSE. Il s'agit de trois composants proposés sous forme de service web pour la gestion de l'authentification, de l'autorisation et du monitoring. Bien évidemment, l'implantation de ces trois services est intégralement basée sur les concepts de sécurité présentés dans les chapitres précédents.
- Enfin dans le chapitre 9, nous présentons nos conclusions ainsi que quelques perspectives par rapport aux problèmes que nous avons traités.

21. Authentification, autorisation et monitoring

Chapitre 2

Problématique et motivations

Sommaire

2.1 Introduction

De nos jours, les réseaux sociaux en ligne (*"Online Social Networks OSN"*) connaissent une utilisation importante, à l'image de *Facebook, Twitter, QQ, Youtube*, *etc*. Par conséquent, de nombreuses recherches se sont focalisées sur ce nouveau cadre conceptuel d'environnements collaboratifs, dont une bonne partie sur l'aspect sécurité, la vie privée et le contrôle d'accès [49, 69, 20, 50, 177, 180, 62, 31, 175]. En effet, le concept d'un réseau social consiste principalement en une plateforme qui fournit des communautés virtuelles pour les gens s'intéressant à un sujet particulier [74, 150] ou juste pour être connectés ensemble afin d'échanger des informations ou des fichiers multimédia. En revanche, de nouveaux enjeux ont attiré l'attention dès que les professionnels se sont intéressés à cette nouvelle tendance de collaboration, à savoir les réseaux sociaux d'entreprise *"RSE"*.

Un réseau social d'entreprise est la variante sociale des environnements collaboratifs qui met l'accent sur la facilité d'échange d'informations et de services, et ce au moyen de plateformes de réseautage social dotées de technologies favorisant la communication et le partage. En effet, à l'image de l'internationalisation des équipes de collaboration, un RSE permet avant tout d'élargir le champ

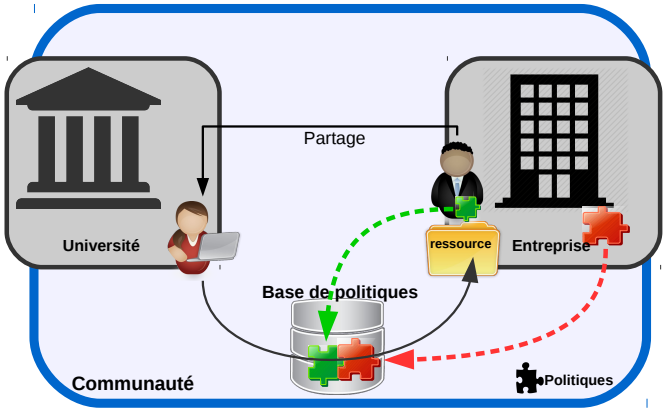

FIGURE 2.1 – Community access control global view.

de collaboration et de compétences. En outre, un RSE permet d'éviter les obstacles d'administration classique [22] qui ralentissent les échanges d'informations et de ressources entre des employés issus de diverses domaines et différentes entreprises.

Cette variante professionnelle des réseaux sociaux, à savoir le RSE, présente de nouveaux enjeux notamment en matière de sécurisation des ressources partagées vis-à-vis d'entités collaboratives externes. Par exemple, nous avons les enjeux économiques et concurrentiels qui requièrent beaucoup de vigilance par rapport à la confidentialité des informations et des ressources partagées.

Dans ce qui suit, nous allons étudier la problématique liée à la sécurisation des échanges collaboratifs dans un contexte RSE. Nous allons en premier lieu aborder l'aspect de conception d'architecture. Nous enchainerons avec la gestion des identités et de l'authentification, car elle constitue la première étape de tout processus de protections de ressources (*i.e.* service **A**AA). Nous allons ensuite nous tourner vers l'aspect contrôle d'accès, et ce dans le but d'analyser les besoins des RSEs en matière de gestion des accréditations d'identités numériques issues de différentes entreprises. Enfin, nous étudierons les besoins pour un mécanisme de contrôle d'accès lui permettront de s'adapter au contexte RSE, et ce dans le cadre de la supervision numérique qui inclut la gestion de la confiance et du risque. Mais d'abord, nous allons présenter une étude de cas pratique illustrant différents scénarios ayant motivés les challenges abordés dans nos travaux de recherche.

2.2 Exemple de motivation

Comme exemple de motivation, nous avons considéré une communauté formée, en premier lieu, d'entreprise de développement de logiciels informatiques et d'une université. La figure 2.1 donne une vue globale sur un scénario typique de collaboration socioprofessionnelle dans lequel un acteur de la société partage un document (*e.g.*, description d'une offre de stage) avec une étudiante de l'université. La ressource à partager est hébergée dans les serveurs de l'entreprise. Nous justifierons

22. Procédures bureaucratiques telles que le Dépôt d'un dossier, la validation par le responsable, le retour, *etc*

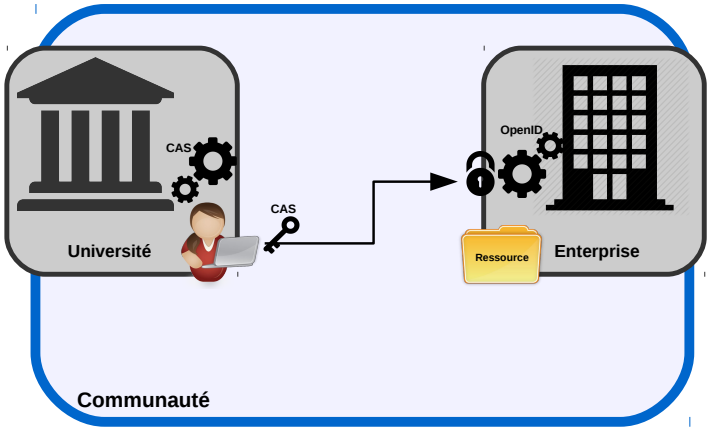

FIGURE 2.2 – Authentification hétérogène dans une communauté.

ce choix plus tard quand nous aborderons l'aspect de conception de l'architecture de collaboration pour *OpenPaaS*.

En premier lieu, nous tenons à élucider à travers un exemple le besoin non fonctionnel concernant l'interopérabilité en matière de gestion d'identité. Comme le montre la figure 2.2, l'entreprise de *James* utilise le mécanisme d'authentification *OpenID*. L'université de *Jessy* utilise un gestionnaire d'identité *CAS*. Il est impératif que l'identité de *Jessy* soit vérifiée et validée auprès du mécanisme d'authentification de l'entreprise de *James* afin que les deux acteurs puissent collaborer ensemble.

Supposons maintenant que la phase d'authentification soit accomplie avec succès, déclenchant ainsi la phase de vérification de privilèges pour ce qui concerne l'identité de *Jessy*. Cependant, dans le domaine universitaire de *Jessy* le terme *"grade"* est utilisé pour faire référence au statut de *Jessy* dans son université, tandis que dans l'entreprise de *James*, le terme utilisé pour faire référence au statut est le *"rôle"*. La question qui se pose dans cette situation est *comment l'entreprise de James va-t-elle considérer le statut de Jessy pour lui attribuer une permission d'accès existante ?* À travers cet exemple, nous essayons de mettre en évidence l'hétérogénéité des communautés de collaboration dans OpenPaaS. En général, le terme hétérogénéité est suivi par le terme *"interopérabilité"*, qui cherche à pallier les problèmes provoqués par l'hétérogénéité.

Pour illustrer davantage nos motivations, cette fois sous l'angle des besoins fonctionnels, nous enrichissons la première communauté avec de nouvelles entités (figure 2.3), en l'occurrence : une entreprise de fourniture de produits alimentaires ainsi qu'une agence de voyage. L'entreprise de

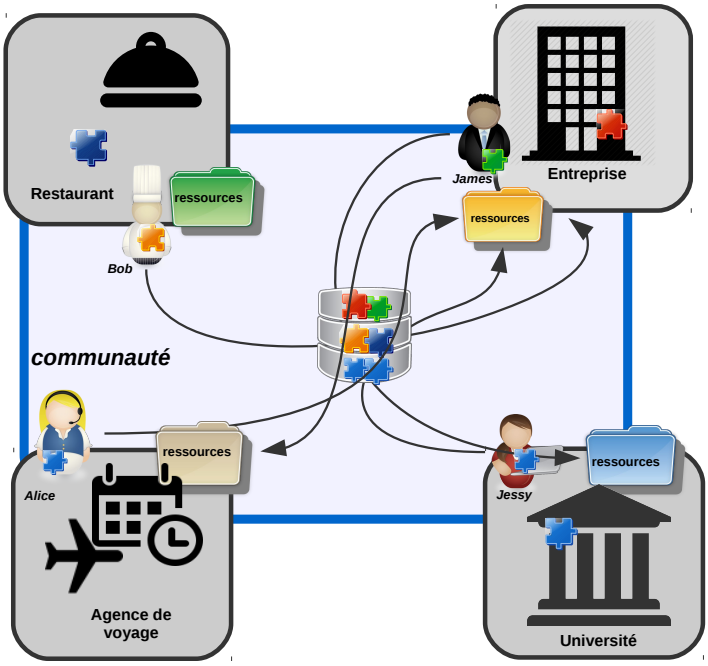

FIGURE 2.3 – Architecture de collaboration vue d'ensemble.

fournitures alimentaires se charge de l'approvisionnement de produits alimentaires (*e.g.*, fruits et légumes) pour les entreprises de la communauté, avec des brochures tarifaires et politiques d'évolution de prix qui diffèrent en fonction de chaque entreprise partenaire. L'agence de voyage se charge de gérer les déplacements du personnel des entreprises partenaires (voyage et séjour) et de proposer évidemment les prix les plus intéressants. L'acteur *Bob* représente l'entreprise d'approvisionnement alimentaire, et l'actrice *Alice* représente l'entreprise de gestion des voyages. Ainsi, au sein de la communauté créée par *James*, ces différents acteurs interagissent en s'échangeant différents types de ressources. Nous envisageons les scénarios suivants.

- Dans un premier scénario, le demandeur d'accès se voit octroyer l'accès à une ressource partagée par les deux parties qui gèrent la ressource, à savoir l'acteur et l'entreprise propriétaires.
- Comme deuxième scénario, nous supposons que le système détecte un comportement suspect. *James* tente d'accéder aux brochures tarifaires proposées par l'entreprise de *Bob* pour les entreprises d'*Alice* et de *Jessy*. On suppose que *James* n'a jamais tenté une telle initiative auparavant. Par conséquent, le système déduit dans un premier temps qu'il s'agit d'une erreur. Cependant, si *James* insiste, le système peut soupçonner qu'il s'est fait usurper son identité par un tiers malveillant, *e.g.*, une autre personne dans l'entreprise.
- Par ailleurs, nous supposons une défaillance du système de contrôle d'accès causée par une ou plusieurs règles obsolètes ou initialement mal définies. Prenons le cas où la sensibilité d'une

ressource collaborative change à travers le temps. Imaginons dans un premier temps que *Carole*, une collègue de *James*, partage un document qui contient du code source important (*i.e.* confidentiel) avec *Jessy* pour son projet. Plus tard, la communauté s'agrandit davantage et plusieurs développeurs logiciel intègrent le projet de *Jessy*. Ainsi, *Carole* constate que la confidentialité et l'intégrité de son document (code source) est en péril. Par conséquent, elle souhaite que certains acteurs soient bannis sur la base de leurs réputations reflétées par leurs comportements respectifs.

- Supposons qu'une autre menace soit révélée par rapport à l'authenticité des attributs d'identités d'un acteur à travers le mécanisme utilisé au niveau de son entreprise pour la gestion de l'authentification. Comme exemple, nous supposons que l'entreprise d'*Alice* utilise comme mécanisme d'authentification le protocole *Login/mot-de-passe*. Supposons que *Alice* parte en vacances et se fasse remplacer par *"Oscar"*. Ainsi, les tâches d'*Alice* seront déléguées à *Oscar*. Cependant, Oscar peut facilement se faire usurper son identité car il utilise un nom d'utilisateur ainsi qu'un mot de passe triviaux.

2.3 Gestion des ressources et identités collaboratives

Avant tout, il est nécessaire d'envisager une conception d'architecture de collaboration pour les communautés du RSE. Cette architecture va influencer la manière dont les différentes entités actives vont pouvoir collaborer et échanger les ressources. Derrière cet échange de ressource s'impose une gestion des identités, des ressources ainsi que des accréditations des identités sur les ressources. Pour cela, deux propriétés sont importantes : le passage à l'échelle et l'autonomie d'administrattion des autorisations sur les ressources déployées.

La conception de l'architecture d'une communauté RSE peut être centralisée, ou bien décentralisée. Chaque type d'architecture présente des avantages et des inconvénients. En effet, nous avons étudié trois types différents d'architecture de collaboration [17] pour la conception de notre plateforme collaborative *OpenPaaS*, à savoir la collaboration à long terme, la collaboration à faible couplage et la collaboration ad-Hoc.

2.3.1 Types de collaboration dans les environnements collaboratifs

Les principaux besoins de conception de plates-formes collaboratives basées sur des infrastructures *Cloud* sont : l'architecture multi-tenants, la virtualisation des ressources, l'administration décentralisée et la fédération des identités. Il existe trois types d'architectures de collaboration [17] qui répondent à ces besoins, à savoir la collaboration à long terme, la collaboration à faible couplage et la collaboration ad-Hoc.

2.3.1.1 Collaboration à long terme

La collaboration à long terme (Fig 2.4) est un type de collaboration basé sur un niveau élevé de dépendances mutuelles et de confiance entre les entreprises. Ce genre de collaboration nécessite

une méta-politique globale compatible avec les politiques propres à chaque entreprise. Ces politiques locales sont intégrées et combinées dans une base de règles commune et centralisée.

La collaboration à long terme offre comme principal avantage la possibilité de mettre en place un cadre collaboratif de longue durée grâce à un ensemble a priori connu d'entreprises et de politiques de collaboration entre ces dernières. De plus, l'aspect centralisé de ce type d'architecture simplifie la gestion des ressources collaboratives, à savoir les identités des acteurs, les ressources, les politiques de contrôle d'accès, *etc*. Cela permet en outre de promouvoir l'interopérabilité en matière de politiques de contrôle d'accès entre les différentes entreprises, car les politiques seront négociées entre tous les partenaires concernant les profils des acteurs et les droits qui s'en suivent dans chaque entreprise.

En revanche, l'architecture de collaboration à long terme présente quelques limites par rapport à notre contexte RSE. En effet, on constate un manque évident de confidentialité et d'intégrité des ressources dû au fait que les entreprises manquent d'autonomie dans la gestion de leurs ressources propres. En effet, les ressources seront hébergées au niveau de la communauté. En outre, dans une collaboration à long terme, établir un plan d'accord concernant les politiques d'échanges collaboratifs entre toutes les entreprises d'une communauté risque de prendre beaucoup de temps, paralysant ainsi le processus de collaboration.

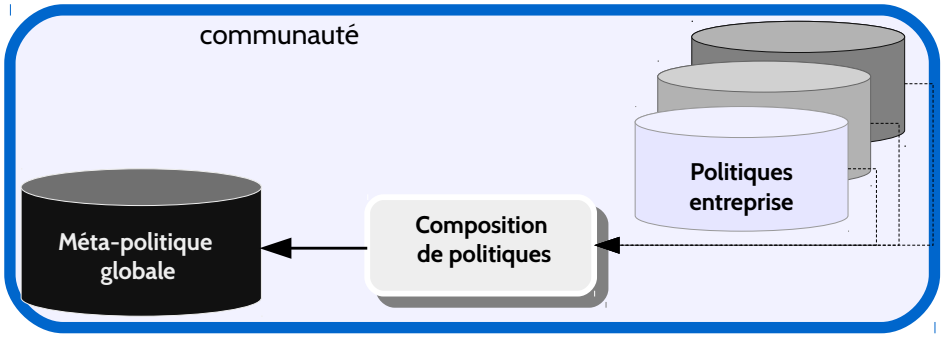

FIGURE 2.4 – Collaboration à long terme [17].

2.3.1.2 Collaboration à faible couplage

Comme son nom l'indique, cette architecture est caractérisée par un couplage faible entre les différentes entreprises dans l'environnement collaboratif. Cela signifie que les politiques locales de contrôle d'accès de chaque entreprise contrôlent l'accès vis-à-vis des interactions avec les entités collaboratrices externes. En effet, comme le montre la figure 2.5, l'accès aux ressources est soumis à deux niveaux de politiques de contrôle d'accès, un niveau de politiques communes interentreprises, et un niveau de politiques indépendantes au niveau de chaque entreprise.

Contrairement à une architecture de collaboration à long terme, cette architecture offre aux

entreprises plus d'indépendance en matière d'administration de politiques. En outre, dans ce type d'architecture, les ressources restent hébergées au niveau des serveurs des entreprises. Par conséquent, une ressource partagée devient accessible via un *Service Level Agreement SLA* assurant une meilleur préservation de l'intégrité et la confidentialité des ressources.

Par rapport à nos objectifs de sécurité, à savoir l'autonomie d'administration de politiques et la faciliter de partage de ressource, nous considérons que l'architecture de collaboration à faible couplage est mieux adaptée à notre contexte RSE que l'architecture de collaboration à long terme. Néanmoins, la gestion indépendante du contrôle d'accès au niveau de chaque entreprise peut provoquer des problèmes d'interopérabilité lors de la confrontation des attributs d'identité d'acteurs externes vis-à-vis des politiques internes de l'entreprise, comme c'est le cas dans le premier exemple de motivation où l'étudiante *Jessy* collabore avec l'entreprise informatique de *James*. Par conséquent, il faudra étudier l'interopérabilité à l'échelle du modèle de contrôle d'accès.

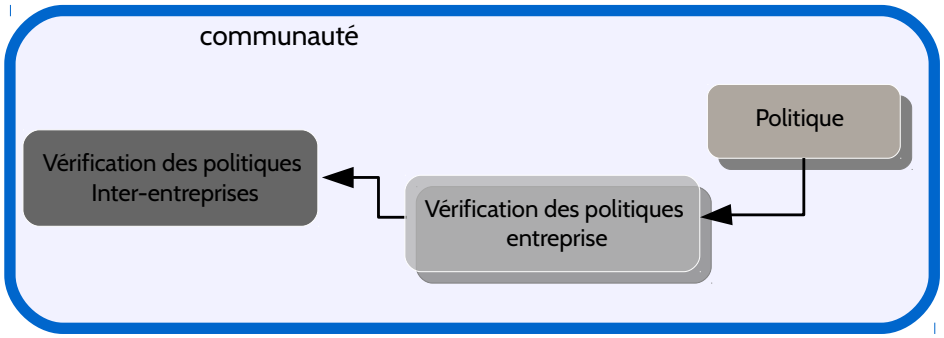

FIGURE 2.5 – Collaboration à faible couplage [17].

2.3.1.3 Collaboration Ad Hoc

Par rapport aux types de collaboration précédents, la collaboration *ad-Hoc* se base sur une infrastructure très flexible, car les durées de collaboration peuvent être éphémères. Par exemple, une entreprise ou un acteur peuvent rejoindre ou quitter une communauté à tout moment. Ainsi, grâce à cette forme de collaboration il n'est pas nécessaire de mettre en place des procédures compliquées pour la gestion de la consommation et/ou du partage de ressources collaboratives. En effet, cela dépend de la fluidité et l'agilité dans la mise en place des politiques de contrôle d'accès. Par exemple, un acteur veut sous traiter une tâche qui lui est confiée à un autre acteur externe. Pour cela, l'acteur principal a besoin de définir les permissions d'une manière simple, flexible et éventuellement temporaire à l'égard de l'acteur qui sous traitera sa tâche. Néanmoins, ce type de collaboration ne favorise pas la collaboration à long terme et donne aux acteurs une autonomie exagérée, ce qui peut parfois être désavantageux pour l'entreprise propriétaire des ressources partagées notamment lorsqu'il s'agit d'acteurs malveillants.

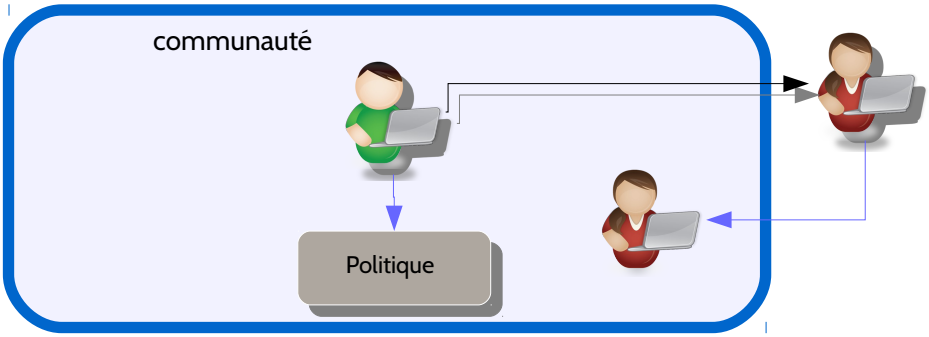

FIGURE 2.6 – Collaboration Ad hoc [17].

2.3.1.4 Synthèse

À l'image du nombre élevé d'utilisateurs et de ressources dans l'environnement ubiquitaire RSE, une gestion totalement centralisée (*i.e.* collaboration à long terme) des ressources ainsi que des accréditations peut avoir un impact négatif sur la qualité de collaboration recherchée par l'utilisation d'un RSE. En effet, cela peut ralentir le processus de collaboration à cause de la complication de l'administration de l'accès aux ressources via des métapolitiques globales. En revanche, une gestion centralisée peut être favorable pour l'interopérabilité en matière de gestion des accréditations et permet également de promouvoir la fédération des identités.

Par ailleurs, à l'image de la sensibilité des ressources professionnelles déployées dans une communauté, une entreprise ne peut pas céder (*i.e.* déléguer) complètement l'administration de ses ressources à une partie tierce, en l'occurrence la communauté RSE. Ainsi, il est primordial pour chaque entreprise de garder son autonomie pour l'administration de ses ressources, *i.e.* collaboration à faible couplage. Néanmoins, cela ne doit pas causer des problèmes d'interopérabilité. En outre, notre conception d'architecture doit également permettre d'établir des collaborations de courte durée, *i.e.* des collaborations ad-Hoc, et ce d'une manière flexible qui n'exige pas une administration compliquée des accréditions sur les ressources collaboratives.

Par conséquent, notre objectif est de trouver un juste milieu entre une approche centralisée et décentralisée tout en respectant les propriétés d'autonomie d'entreprise, de fluidité d'échanges collaboratifs et d'interopérabilité.

Nous allons par la suite nous intéressé à la gestion de l'authentification des identités numériques au sein des communautés RSE.

2.3.2 Gestion de l'authentification

Dans le cadre de notre conception du modèle de sécurité pour le RSE OpenPaaS, nous considérons que les communautés de collaboration sont indépendantes en matière de gestion des identités et des accréditations. Cela signifie d'un côté qu'un acteur dans une communauté donnée ne peut

pas hériter des permissions depuis d'autres communautés, et d'un autre côté cela veut dire que les ressources ne sont pas transférables d'une communauté à d'autres. En d'autres termes, les identités ainsi que les permissions sont *mono-communautaire*. Cela signifie qu'une phase d'authentification est indispensable au niveau de chaque communauté, car elle permet de lier chaque identité à ses droits d'accès définis dans la communauté en question.

Dans un cadre RSE, chaque entreprise dispose en réalité de son propre mécanisme de gestion d'identités numériques qu'elle voudrait naturellement préserver au sein du RSE. Un mécanisme de gestion d'identité utilise un protocole d'authentification qui peut différer d'une entreprise à une autre, *e.g.* login/mot-de-passe, Pin, OpenID, *etc*. Cela signifie que, pour qu'un acteur puisse accéder à une ressource partagée, son identité doit être approuvée par le mécanisme de gestion des identités de l'entreprise externe propriétaire de la ressource, et ce en fonction du protocole d'authentification utilisé par cette dernière. À l'image de l'hétérogénéité des protocoles d'authentification existants [23], la principale question qui se pose dans ce contexte est relative à la manière dont sera traitée l'interopérabilité (en matière d'authentification) au sein de la même communauté. Il n'est en effet pas évident de concevoir une plate-forme qui permet de gérer cette interopérabilité. Par conséquent, il convient de trouver une solution alternative qui permette de gérer cette interopérabilité.

2.4 Contrôle d'accès

L'idée de collaboration au sein d'une communauté *OpenPaaS* est basée sur l'échange entre au moins deux entreprises. Le choix d'une démarche de collaboration basée sur le concept de réseaux sociaux vise en premier lieu à rendre les échanges entre collaborateurs plus fluides, *i.e.* plus de facilité dans la procédure de partage de ressource. Étant indépendante, chaque entreprise dispose de ses propres politiques de gestion de l'accès à ses ressources, y compris celles destinées à la collaboration. Cela signifie qu'il est nécessaire de mettre en place un modèle de politiques collaboratives qui permet d'homogénéiser les différentes politiques des entreprises. Par conséquent, afin de négocier les droits par rapport à la manière dont chaque entreprise attribue les privilèges à chaque caractéristique identitaire, le modèle de politiques collaboratives doit être flexible. Par exemple, dans une université, le statut *"chef de département"* donne le droit à l'entité qui le possède de consulter librement l'avancement de tous les projets de stages des étudiants de son département. Tandis qu'un chef de département dans une entreprise (avec qui l'université collabore) ne peut pas avoir un tel privilège, seul le chef de département de l'université dispose d'un tel droit. Ainsi, la caractéristique chef de département dans l'entreprise doit être mixée avec l'attribut qui le qualifie de *"superviseur"* de projet de stage pour qu'il puisse suivre l'avancement du projet de l'étudiant en question. En réalité, de telles négociations ne sont pas faciles à établir et prennent dans certain cas beaucoup de temps (dépassant parfois le délai prévu pour la réalisation du projet collaboratif). Dans ce cas, cela signifie que le RSE n'apporte plus un grand intérêt en matière de collaboration. Par conséquent, il est peut-être plus ju-

23. La plupart des protocoles d'authentification existants sont codés de manières très différentes et sont souvent incompatibles les uns avec autres. Par exemple, un jeton d'authentification conforme au protocole *SAML* n'est pas compatible avec un mécanisme utilisant le protocole *OAuth* ou *OpenID*

dicieux de se servir de l'agilité (offerte par le RSE) en matière de partage de ressources et de confier ainsi la tâche de définition de droits aux entités directement impliquées (d'une manière active) dans la collaboration, en l'occurrence les principaux sujets de collaboration, à savoir les acteurs humains.

Afin qu'un acteur puisse définir une règle de contrôle d'accès, il est nécessaire que cette dernière soit simple. Plus précisément, à partir du moment où un acteur sait avec qui il doit partager une ressource donnée et de quelle manière (action autorisée), il peut définir ce droit en précisant seulement le sujet, l'objet (la ressource) et l'action (lecture/ écriture/ exécution). En effet, un acteur ne peut pas définir des règles globales telle que celle présentée dans l'exemple précédent (**cf.** chef de département). Dans un contexte RSE, cela a en revanche un contrecoup relatif à une utilisation abusive d'une telle autonomie par l'acteur. En effet, l'acteur est libre de définir des droits sur des ressources appartenant à son entreprise comme bon lui semble. De plus, même un acteur bienveillant peut se tromper dans la définition d'une règle donnant le droit d'accès à une mauvaise entité active (un concurrent économique par exemple). Par conséquent, l'entreprise doit garder un contrôle sur les règles définies par ses acteurs. Cependant, en raison de la grande fréquence de partage dans une communauté RSE, le contrôle de l'entreprise ne peut pas être au même niveau de granularité que celui des acteurs. En d'autres termes, une entreprise ne peut pas contrôler chaque nouvelle règle définie au sein de chaque communauté. Nous reviendrons sur cet aspect avec davantage de détails quand nous aborderons la problématique liée à la supervision (**cf. Section 2.5**).

Un autre souci lié à l'administration de règles basées sur les acteurs concerne les éventuelles incohérences dans la définition de la règle. En d'autres termes, un acteur peut mal définir une règle de contrôle d'accès, en se trompant ou en oubliant un des attributs de la cible (sujet/objet/action). Par conséquent, il est primordial de vérifier que les règles soient bien définies. Une telle vérification peut être réalisée de différentes manières, elle peut être manuelle ou automatique. Cependant, cette vérification est dans les deux cas très coûteuse. Par exemple, il est très fréquent dans le cas des règles définies à base d'une syntaxe *XML (e.g., XACML)* que les utilisateurs définissent mal des règles de contrôle d'accès. Une procédure de vérification est possible au moyen des modules de vérification au niveau client. Sachant que nous sommes dans un environnement ubiquitaire (RSE), une alternative moins coûteuse qui permet de palier le problème de la vérification automatique est d'opter pour un formalisme logique en guise de langage de définition de règle. En dépit de la difficulté [24] d'écrire des prédicats logiques, un langage formel doté d'un raisonneur automatique permet de résoudre le problème de vérification et de réduire considérablement les coûts [25] de vérification de la cohérence.

Par ailleurs, dans un RSE, une ressource partagée entre deux ou plusieurs acteurs est une copropriété de tous ces acteurs. Par conséquent, il est important qu'un acteur donné puisse revendiquer (pour une raison donnée) l'accès sur une ressource partagée par un de ses copropriétaires. Cela implique le besoin de combinaison de plusieurs règles définies par des acteurs à l'égard d'une même ressource, et gérer par conséquent les possibles incohérences entre règles.

L'idée de faire établir des règles d'autorisation par des acteurs collaboratifs favorise en outre le

24. Que nous avons banalisée grâce à un programme de génération automatique qui se base sur un ensemble d'attributs, en l'occurrence la cible.

25. Une illustration des performances sera présentée dans le chapitre **cd. Mise en œuvre**.

mode de collaboration ad-Hoc. En effet, une collaboration ad-Hoc peut être exploitée afin de permettre à un acteur de sous traiter des tâches à d'autres acteurs en s'appuyant sur la facilité de définition d'autorisations. Une sous traitance de tâche peut être vue comme une *délégation* [204, 79, 167, 197] de droits d'un délégataire à un délégué afin qu'il puisse accéder à ces ressources. Cependant, une collaboration ad-Hoc est éphémère et les droits d'accès le sont également. Par conséquent, la règle d'autorisation en question doit être supprimée à la fin de la collaboration. Néanmoins, quand il s'agit d'acteurs humains (la majorité des acteurs d'un RSE en général) le risque d'oubli peut avoir un impact négatif sur la confidentialité des ressources déléguées. Ainsi, une bonne solution est d'automatiser la suspension de la règle après une période de validation prévue. Cela implique d'inclure l'aspect temporel dans la définition de la règle. Définir des contraintes temporelles sur une règle de contrôle d'accès ne constitue pas en-soi une tâche très compliquée. C'est le cas, par exemple, dans les politiques XACML où de telles contraintes temporelles peuvent être définies à base de *XML*. Cependant, dans la perspective de tirer profit de l'avantage de la vérification automatique de la cohérence des règles que nous fournit un langage formel, il reste difficile d'inclure le temps dans la modélisation de règles.

2.5 Supervision et confiance numérique

Dans le cadre de notre discussion à propos de la conception d'un modèle de règles qui soient définies par les acteurs de collaborations, nous avons souligné l'importance de la vérification de ces règles par les entreprises concernées. En effet, le contrôle se fait plutôt vis-à-vis des requêtes externes, car on se base sur l'hypothèse que l'acteur interne ne soit pas corrompu (ou compromis) et risque seulement de se tromper au moment de la définition de la règle. En d'autres termes, on s'intéresse au profil dynamique du sujet de la requête appartenant à une entreprise partenaire. Un profil dynamique d'un acteur de collaboration est représenté par certains attributs susceptibles de changer de valeurs à travers le temps [26], notamment ceux qui représentent son comportement comme le nombre de ses tentatives d'accès, le temps passé par session, la fréquence de partage de ressources, *etc*.

En réalité, un acteur peut avoir différents comportements pendant ses expériences collaboratives dont certains comportements peuvent constituer des menaces de sécurité pour les entreprises, *i.e.* comportement malveillant. De nombreuses approches dans différents travaux [107, 149, 46, 183, 47] ont essayé de prédire le comportement futur d'un acteur. Néanmoins, le comportement d'une entité active reste imprévisible et peut parfois s'avérer très variable et instable, notamment dans le cadre des réseaux sociaux d'entreprise où les acteurs sont majoritairement des humains. Ainsi, à l'image de l'enjeu majeur de la confidentialité des ressources professionnelles, cela risque de causer de sérieux dommages pour les entreprises.

Pour faire face à l'aspect dynamique du comportement d'un acteur humain dans un contexte collaboratif, la question se pose naturellement par rapport à la *confiance* qu'un système de contrôle d'accès peut accorder à cet acteur. C'est pourquoi, le comportement de tout acteur doit être en permanence supervisé et ses traces d'interactions collaboratives enregistrées et archivées. La supervision à l'égard d'un acteur est généralement basée sur l'exploitation de l'historique d'interaction enregistré. Sous le volet du contrôle d'accès, cette analyse se doit d'être pertinente afin de promouvoir la fiabilité des décisions d'autorisations d'accès prises au niveau des communautés RSE. En outre, elle permettra de remettre en cause certaines règles (pour une éventuelle révision) jugées obsolètes vis-à-vis de tout acteur témoignant d'une mauvaise réputation.

Selon notre point de vue, une analyse pertinente de l'historique comportemental d'un acteur doit permettre d'agir rapidement suite à l'interception d'un comportement douteux. Agir dans notre contexte est synonyme de rejeter la requête et par conséquent pénaliser l'acteur qui en est à l'origine par rapport à sa réputation [27] dans le cadre collaboratif. Par ailleurs, à l'image du cadre social de la vie réelle, la confiance doit pouvoir s'améliorer suite au maintien de comportements corrects. Dans la même optique, la vitesse d'amélioration de la confiance est loin d'être aussi rapide que sa dégradation. En résumé, dans le but de répondre à ces besoins, il convient d'intégrer dans notre plate-forme un système *dynamique* d'évaluation de la confiance de chaque acteur suivant l'évolution de sa réputation sur une ligne temporelle continue. Pour chaque sujet, la réputation et ainsi le niveau

26. Les informations statiques telles que (*le nom, l'age, le rôle, etc.*) sont ignorées.
27. La réputation a un impact direct sur l'évolution positive ou négative de la confiance d'un acteur, **cf.** chapitre **Confiance numérique**

de confiance peuvent évoluer d'une manière continue, positivement ou négativement, en fonction du comportement du sujet à travers ses sessions collaboratives. Cependant, la question qui reste posée est : *comment une évaluation de la confiance peut-elle servir les entreprises pour mettre en place des politiques abstraites et efficaces ?*

2.5.1 Gestion du risque des requêtes de demande d'accès

Dans le cadre de la supervision et l'évaluation de la confiance, nous avons souligné l'importance du besoin de la prise en considération du comportement de chaque entité active, principalement les acteurs humains. Par conséquent, nous avons opté pour un mécanisme de gestion du risque pour étudier la possibilité d'exploiter et d'intégrer la confiance, avec éventuellement de nouvelles variables liées au contexte RSE.

Outre les réputations des acteurs externes, d'autres variables entrent en considération, comme l'importance d'une ressource collaborative qui peut changer à travers le temps en fonction du nombre d'acteurs concernés. Par ailleurs, l'hétérogénéité d'une communauté en matière d'authentification d'acteurs peut laisser place à certaines vulnérabilités, dans la mesure où une entreprise ne peut pas évaluer la fiabilité d'un mécanisme d'authentification utilisé par une autre entreprise partenaire.

Une ressource dans un RSE peut appartenir à plusieurs propriétaires en même temps. Le nombre de propriétaires peut également changer à travers le temps. Ce changement est susceptible de provoquer une altération de l'importance de la ressource en question, et ce en partant du principe que, l'importance d'une ressource collaborative augmente avec l'accroissement du nombre de propriétaires. Ainsi, ce critère d'importance peut également être très important pour la phase de contrôle des politiques de partage au niveau des entreprises.

Afin de promouvoir la collaboration et respecter les préférences des entreprises en matière de gestion d'authentification, nous avons souligné le fait qu'au sein d'une communauté on ne doit pas imposer un mécanisme commun à toutes les entreprises. Dans l'hypothèse où on arrive à gérer cette interopérabilité, il reste à étudier les contrecoups d'une telle hétérogénéité sur la gestion des autorisations d'accès aux ressources collaboratives vu que les deux procédures, à savoir l'authentification et l'autorisation, sont complémentaires. En effet, tous les mécanismes d'authentification ne reflètent pas le même niveau d'intégrité, ni de certification de l'authenticité des identités numériques authentifiées. Ainsi, la vulnérabilité de l'environnement collaboratif, à savoir la communauté, peut être liée à la fiabilité des identités en matière d'authenticité.

2.6 Synthèse

Cette section résume la problématique de cette thèse qui porte sur la sécurisation des échanges collaboratifs au sein d'un RSE.

L'hypothèse générale qui peut résumer la problématique est la suivante : *"un acteur obtient un accès à une ressource qui lui est normalement non autorisé et met ainsi en péril la confidentialité de la ressource"*. En effet, même avec l'utilisation d'un système de contrôle d'accès classique, plusieurs

scénarios menaçants peuvent se produire, tels que :

- La règle de contrôle d'accès a été initialement mal définie,
- un pirate informatique usurpe l'identité d'un utilisateur de confiance ou un utilisateur existant devient malveillant et essaie de voler des informations précieuses,
- la sensibilité (*i.e.* l'importance) d'une ressource collaborative donnée change à travers le temps, ce qui rend la règle de contrôle d'accès initiale obsolète.
- le mécanisme d'authentification utilisé pour l'authentification de l'identité du demandeur d'accès inclut des failles de sécurité, ce qui signifie que l'authenticité de l'identité de l'acteur en question ne peut pas être garantie.

Ceci nous conduit à relever, pour la conception du mécanisme de contrôle d'accès destiné au RSE *OpenPaaS*, les besoins suivants :

- la flexibilité du modèle de règle afin de gérer l'interopérabilité entre les différentes entreprises collaboratives en matière de gouvernance du contrôle d'accès,
- la possibilité de définir les règles négatives (*i.e.* interdiction) afin de permettre à un acteur de revendiquer l'accès à une ressources donnée (*i.e.* décentralisation du pouvoir de gestion de ressources collaboratives).
- l'autonomie de l'entreprise par la possibilité de filtrer les règles définies par ses acteurs, et ce, avec un niveau assez élevé d'abstraction,
- la considération du contexte dynamique de collaboration,
- la prise en considération du temps dans la définition des règles de délégations auto-révocables (*i.e.* permissions temporaires),
- la vérification automatique et non coûteuse de la consistance et la cohérence des politiques présentes en très grande masse dans les communautés RSE.

2.7 Conclusion

Dans ces deux premiers chapitres (*Introduction générale* et *Problématique et motivations*), nous avons introduit nos travaux réalisés dans le cadre de cette thèse. D'abord nous avons présenté le contexte ainsi que les objectifs sur lesquels nous nous sommes focalisés pour étudier les différents aspects de problématique ayant guidé nos différentes contributions. Dans les chapitres suivants, nous allons présenter ces contributions, mais avant nous introduisant l'*État de l'art* qui permettra de mieux comprendre certains de nos choix et aidera à la compréhension de nos solutions.

Chapitre 3

État de l'art

Sommaire

Dans ce chapitre nous allons présenter un état de l'art portant sur des travaux ayant été réalisés dans des contextes similaires au notre, à savoir la gestion des identités numérique et des autorisations de contrôle d'accès dans les environnements collaboratifs. Nous présenterons par ailleurs des concepts que nous avons utilisés dans l'élaboration de nos solutions en la matière.

3.1 Gestion des identités numériques collaboratives

Dans un environnement collaboratif, il existe deux types d'entité de collaboration, à savoir les entités passives et les entités actives. Une entité active peut être un utilisateur humain, ou un programme informatique (un service web, un agent informatique, *etc.*). Quant à l'entité passive, elle peut être tout objet utilisable par une entité active comme une ressource, un logiciel, une information, *etc*. La principale différence en termes de sécurité est que, contrairement à une entité active, une entité passive peut uniquement recevoir des informations, mais ne peut pas les revendiquer auprès d'autres entités (passives ou actives).

Toute entité collaborative (processus, services, données, mais aussi utilisateurs, logiciels) possède une identité numérique la distinguant au sein de chaque communauté de collaboration. Une entité est caractérisée par un ensemble d'attributs permettant de déterminer son profil dans le cadre collaboratif (confiance, comportement, rôle, sensibilité, *etc.*). Une bonne gestion des identités numériques doit prendre en considération deux aspects fondamentaux, à savoir leurs certifications (***authentification***) et leurs accréditations d'accès (***autorisation***). Dans cette section, nous nous sommes focalisés sur la partie *authentification* et la gestion des identités numériques.

3.1.1 Identité et cycle de vie

Bien que l'identité d'une quelconque entité est censée être indépendante et unique, elle est en réalité toujours reliée à un ***domaine*** faisant partie d'un contexte spécifique. Par exemple : l'identité *"Bob"* dans le contexte social n'est pas la même que *"Mr. Bernard"* dans le domaine professionnel. Cependant, ces deux identités font référence la même personne. Par conséquent, l'identité de chaque entité active doit rester unique, éventuellement avec de multiples références pouvant la distinguer dans chacun des domaines de référence.

Une entité est donc caractérisée par un ou plusieurs attributs ayant une sémantique particulière par rapport à un domaine. L'ensemble des attributs d'une entité quelconque est accessible via un identifiant interne *i.e. identifiant privée au domaine*. La figure 3.1 donne une vue globale sur les relations domaine-entité-identité.

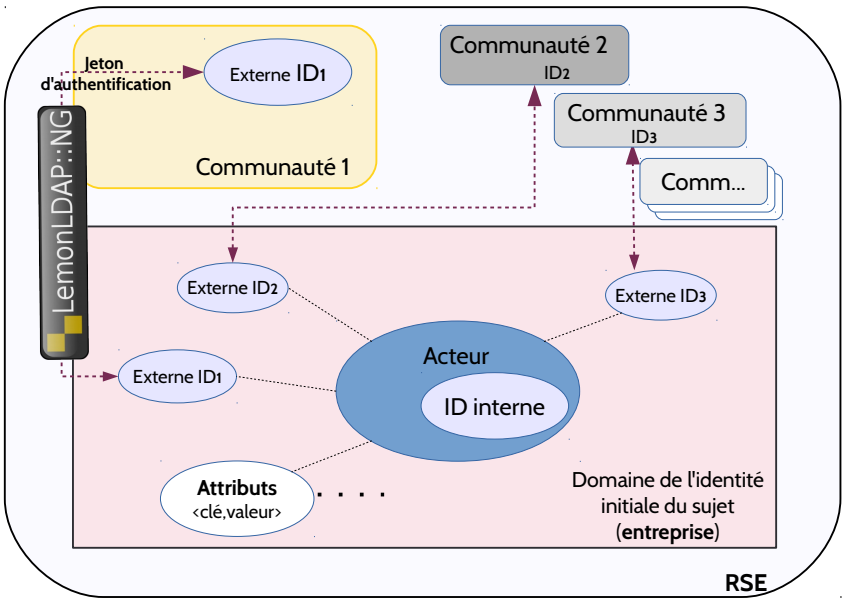

FIGURE 3.1 – Gestion des identité dans les communautés fédérées.

L'identifiant interne (ID interne) représente l'identité initiale d'une entité dans son domaine initial. Un domaine initial d'identité est le cadre dans lequel l'identité de l'entité en question a été

initialement créée. L'identifiant interne présente quelques particularités et exigences par rapport aux autres attributs d'identité, à savoir :

- l'identifiant interne permet de distinguer l'entité en question par rapport aux autres entités du même domaine (*i.e.* les identités internes) ;
- l'identifiant interne doit être indépendant de la sémantique que peuvent avoir les attributs de l'entité qu'il représente ;
- l'identifiant interne ne doit pas avoir un sens particulier ou être relié à d'autres attributs ;
- l'identifiant interne est restreint (privé) au domaine de l'entité qu'il représente, il ne doit jamais être visible à partir d'autres domaines externes.

Cependant, dans un cadre collaboratif une entité doit pouvoir avoir une identité externe à son domaine initial. En d'autres termes, une entité collaborative active a besoin d'un identifiant qui soit utilisable dans les autres domaines avec lesquels elle souhaite collaborer. On parle dans ce cas d'**identifiant externe** qui sert à définir une facette d'identité adaptée à un domaine donné à partir d'une identité initiale. En d'autres mots, l'identifiant externe peut être lié à un sous-ensemble de l'ensemble intégral des attributs identitaires stockés dans le domaine initial de l'entité en question. Cela permet d'avoir une bonne flexibilité dans la collaboration (faciliter de définir des identités) tout en ayant une bonne confidentialité des attributs identitaires. Par exemple, dans une communauté d'entraide en programmation informatique, un acteur n'a pas besoin de divulguer certains attributs d'ordre personnel, comme son numéro de téléphone, adresse résidentielle ou numéro de sécurité sociale.

D'un point de vue protection de ressources collaboratives, l'*identifiant externe* sert à authentifier une entité dans un domaine qui lui est externe. Plusieurs méthodes sont utilisées pour une procédure d'authentification, à savoir des méthodes centralisées et des méthodes distribuées.

3.1.2 Étude des solutions existantes

Avec l'évolution des architectures des systèmes, les méthodes d'authentification sont passées de méthodes classiques et centralisées, telles que le fameux *login/mot-de-passe* et outils similaires, à des méthodes décentralisées davantage orientées vers les environnements multi-domaines.

3.1.2.1 Authentification forte

L'authentification forte (**figure**.3.2) est une nouvelle famille des protocoles d'authentification classiques basées sur de multiples dispositifs d'authentification comme la mémoire de l'utilisateur (Login/mot-de-passe, code pin, *etc.*), les systèmes de codage aléatoire (envoi d'un sms sur le téléphone portable personnel), ou les procédés biométriques (empreinte digitale, reconnaissance vocale/rétinienne, *etc.*).

L'authentification forte est une nouvelle forme de vérification des identités numérique qui se base sur la facilité de l'accès aux informations à tout moment [28] pour renforcer les procédures classiques

28. *e.g.* la lecture d'un sms ou d'un e-mail depuis son smartphone

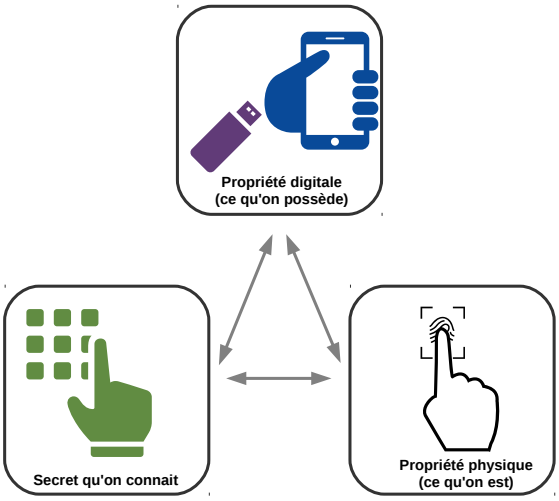

FIGURE 3.2 – Authentification forte

basées sur des informations secrètes fournies par l'utilisateur final (*e.g.* un mot de passe), et ce en étendant le processus sur deux phases de vérifications.

Avec les outils récents de génération de mots de passe, ces derniers sont devenus assez difficiles à déduire (*e.g* WPA, WEP TKIP, WEP TKIP+AES, WPA2), cependant ils restent vulnérables. Car un mot de passe peut être piraté ou bien juste volé (observé) par manque de vigilance du possesseur. Or, avec l'authentification forte, l'utilisateur (à authentifier) doit justifier de la possession d'un code à usage unique envoyé sur son adresse mail ou sur son téléphone mobile sous forme de message texte, et ce sur un terminal qu'il avait lui même associé à son compte au préalable. Le concept d'authentification forte (ou la validation en deux étapes) a été développé et adopté par des compagnies telles que *Google, Paypal, Dropbox, Twitter*. *FIDO* [77] est un système qui a très bien réussi la mise en œuvre du concept de l'authentification forte.

3.1.2.2 Authentification unique multi-domaine (SSO)

Dans une communauté RSE qui regroupe plusieurs entreprises, l'identité d'un acteur doit être validée auprès de chaque entreprise qui possède des ressources qui intéressent l'acteur. Cela signifie que l'acteur en question doit s'authentifier plusieurs fois avec la même identité, ce qui constitue une redondance qui peut nuire à la motivation collaborative de l'acteur en question. Ainsi, il est plus pratique pour un acteur de s'authentifier une seule fois afin d'accéder à plusieurs ressources (plusieurs fois) tout en s'épargnant de multiples procédures d'authentification. Cette dernière configuration est l'idée fondatrice du concept de l'authentification unique *"SSO" **S**ingle **S**ign **O**n*. Parmi les protocoles SSO les plus connus, on distingue le protocole OpenID, le protocole OAuth et le protocole SAML.

OpenID

Le protocole *OpenID 2.0* [162, 110] fournit à un utilisateur un identifiant *OpenID* spécifique lui permettant de s'identifier auprès des sites compatibles, *i.e.* supportant le protocole. Un utilisateur peut créer lui même son propre *OpenID* et le mettre sur un serveur vers lequel seront redirigés les services consommateurs [29]. Il peut également utiliser des fournisseurs d'identités *OpenID* existants comme *Yahoo, AOL, Google, Orange, etc.*

La partie droite de la figure 3.3 illustre le mode de fonctionnement du protocole *OpenID*. La principale spécificité réside dans le fait que l'application du côté client (navigateur web) se charge de confirmer l'authenticité de l'utilisateur vis-à-vis du fournisseur de services à chaque fois que ce dernier le réclame, et ce par le biais de cookies [30].

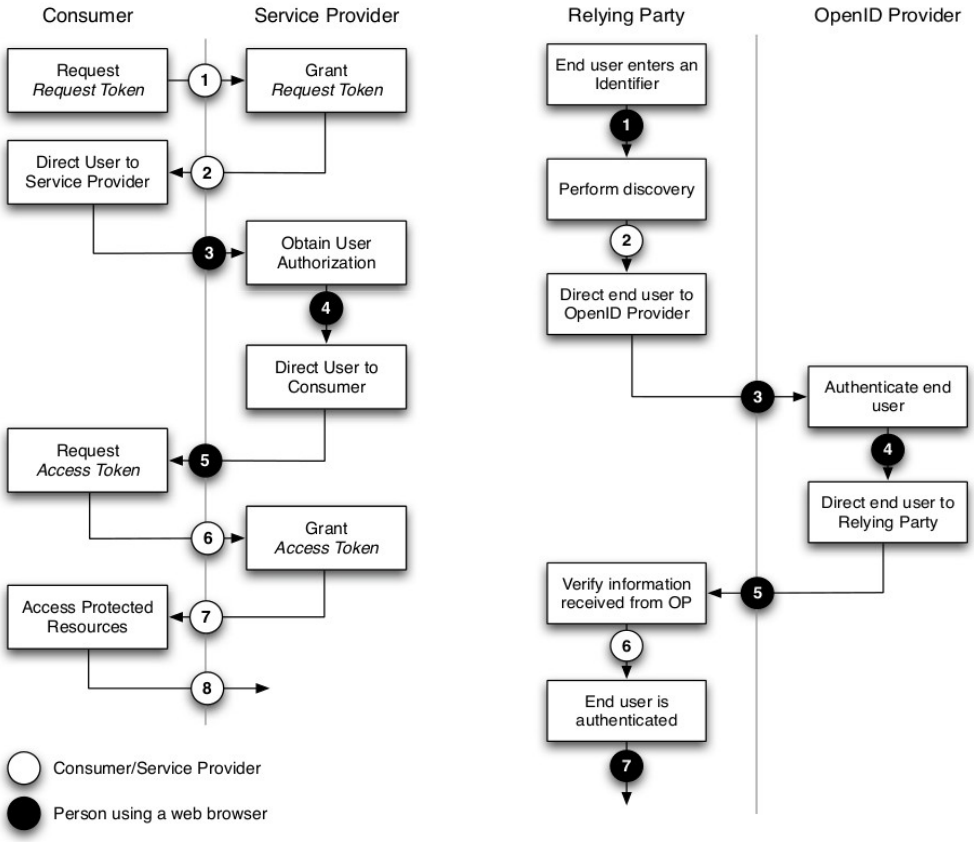

FIGURE 3.3 – OAuth/OpenID authentification [110].

29. Les utilisateurs, souhaitant élaborer de manière autonome leur propre OpenID, peuvent se contenter d'un simple fichier texte dans lequel sont enregistrées les informations concernant leur identité.
30. https ://tools.ietf.org/

OAuth

OAuth 1.0 est un protocole d'autorisation standard ouvert qui permet à des tiers (principalement des applications et service web) d'accéder aux données des utilisateurs sans connaître leur mot de passe [10]. *OAuth* peut être considéré comme un protocole de délégation identitaire qui donne à une application tierce le droit d'agir au nom d'un utilisateur sur la base d'un ensemble (éventuellement restreint) de ses attributs identitaires. En effet, *OAuth* permet de notifier un fournisseur de ressources (par exemple *Twitter*) que le propriétaire de la ressource accorde la permission à un tiers (par exemple, une application de *e-commerce*) d'accéder à ses informations (par exemple, la liste de contacts). La particularité de *OAuth* est qu'il est adapté aux services qui ne sont pas forcément des applications web, comme des applications bureautiques, des appareils mobiles, des *set-top box* (Box ou décodeur TV)[9].

Le processus d'autorisation se base sur une requête qui redirige le propriétaire vers son fournisseur OAuth où sont stockées les données à utiliser. Le mode de fonctionnement du protocole OAuth est illustré sur la partie gauche de la figure 3.3.

Discussion

OAuth et *OpenID* partagent beaucoup de propriétés à savoir la gestion d'identité, la décentralisation, la redirection entre sites. La principale différence entre *OAuth* et *OpenID* est que *OAuth* permet à un tiers d'accéder à des données protégées avec une granularité contrôlée par le propriétaire de ces données.

La question qui résume le principal souci avec l'utilisation des protocoles *SSO* est *"À quel point l'utilisateur peut confier ses informations à un fournisseur SSO ?"* De plus, du point de vue du fournisseur de services, les protocoles SSO (et en particulier *OpenID*) ne présentent aucune garantie sur l'authenticité des informations sur l'utilisateur à authentifier. En outre, la redirection entre plusieurs sites expose l'utilisateur au risque de tiers malveillants, *e.g. le Phishing*. Ainsi, un cadre de confiance est nécessaire pour éviter les limites des protocoles SSO. Dans la section suivante, nous présenterons la notion de **fédération** qui permet de réduire le risque d'interaction avec les tiers malveillants tout en préservant l'agilité en matière de procédure d'authentification offerte par les approches **SSO**.

3.1.2.3 Authentification fédérée

Un des problèmes de l'authentification unique concerne le manque de garantie concernant l'authenticité des informations d'identité communiquées par le fournisseur de l'identité. Le problème peut avoir un impact négatif sur les deux parties, le client SSO (*i.e.* le fournisseur de services) et l'utilisateur. L'utilisateur risque une utilisation abusive de ses informations par son fournisseur SSO. Par ailleurs, le client SSO ne peut pas s'assurer de l'identité fournie car **il ne connaît pas le fournisseur** de cette dernière. Pour remédier à ces problèmes, une solution peut être la mise en place d'une **fédération** qui peut être considérée comme un périmètre de confiance mutuelle entre domaines (*i.e.* entreprises). Les conventions entre partenaires peuvent être établies au moyen de politiques per-

mettant d'encadrer les interactions entre membres et par ailleurs d'installer un cadre de confiance notamment concernant les informations identitaires inter-domaines. Ainsi, la fédération [80] est une approche très intéressante pour faciliter et certifier le partage d'informations dans un environnement collaboratif hétérogène tel qu'un RSE. Un exemple de gestion des identités fédérées est le standard *SAML* [137], proposé par *OASIS*.

SAML

Développé par OASIS [31], *SAML* (Security Assertions Mark-up Language) est un standard de fédération d'identité basé sur le langage *XML*. Comme illustré sur la figure 3.4, *SAML* permet d'établir un pont entre un acteur, un fournisseur de services et un fournisseur d'identité. L'acteur demande d'accès à une ressource auprès du fournisseur de services, ce dernier exige une certification de l'identité de l'acteur dont le protocole *SAML* se chargera de lui fournir. Pour cela, l'acteur doit d'abord s'authentifier auprès de son fournisseur d'identité. Une fois la phase d'authentification de l'acteur auprès de son fournisseur d'authentification réussie, *SAML* transfère au fournisseur de services une affirmation concernant l'authenticité identité de l'acteur en question, et ce quel que soit le protocole d'authentification utilisé par le fournisseur d'identité. Plus précisément, *SAML* n'indique pas au fournisseur de services le protocole d'authentification utilisé. En résumé, *SAML* est comme un contrat d'assurance entre différents partenaires, d'où la définition de la notion de fédération. Cependant, le fait de cacher le mécanisme utilisé pour l'authentification peut nuire à la qualité de la collaboration dans un environnement RSE dans le sens où une telle information est nécessaire afin de juger l'intégrité des entités avec lesquelles une entreprise collabore. Par exemple, une entreprise peut décider de ne pas collaborer avec les entreprises utilisant le système *Login/mot-de-passe* car elle le considère comme étant très vulnérable.

Diverses normes de fédération sont basées sur le standard SAML, par exemple *Liberty Alliance ID-FF, ID-WSF,* et *WS-Federation*.

WS-Federation

WS-Federation est une des normes de fédération basée sur **SAML** présentée par Microsoft et IBM dans l'article [57] en 2002, décrivant un guide pour l'élaboration d'un ensemble de spécifications de sécurité des services-Web incluant **WS-Security, WS-Policy, WS-Trust, WS-Federation, WS-Privacy, WS-Authorization et WS-SecureConversation**.

WS-Security est le cadre global qui définit les fonctions de base pour assurer l'authenticité l'intégrité et la confidentialité des messages, en se basant sur l'utilisation de jetons de sécurité. Afin d'échanger des messages de manière sécurisée, WS-SecurityPolicy permet d'établir la description des exigences de sécurité, et ce par l'évaluation du type de jetons acceptés.

WS-Trust est le service fondamental des fédérations du type WS-federation. Il gère la gestion des jetons de sécurité. Il définit des protocoles pour délivrer, renouveler et annuler des jetons WS-

31. Organization for the Advancement of Structured Information Standards (OASIS)" est un consortium mondial à but non lucratif, fondé en 1993. Il a conduit le développement et l'adoption de normes (standards) de commerce électronique.

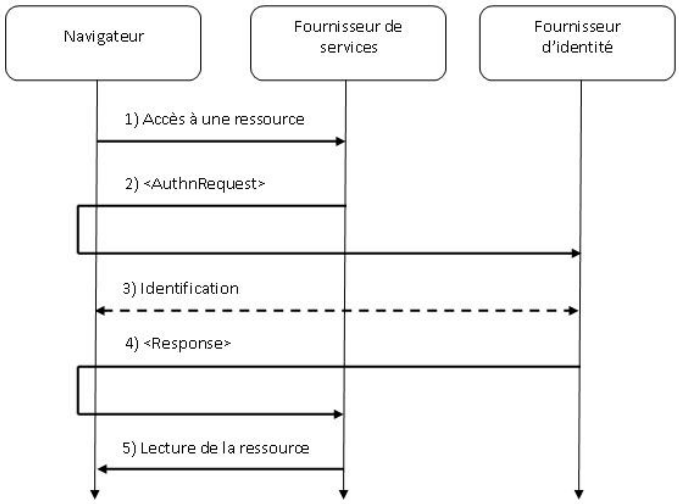

FIGURE 3.4 – SAML

Security, et ce au moyen d'échange de messages sécurisés à travers des services web. Pour assurer la gestion de messages entre parties distribuées, WS-Trust se base sur un module d'échange de jetons (sous forme de requête/réponse) [4] appelé *Security Token Service* (STS). Ces jetons de sécurité sont décrits par *WS-SecurityPolicy* et utilisés par *WS-Security*.

En effet, un STS peut être considéré comme le garant ou l'intermédiaire de la relation de confiance entre les différentes parties d'une interaction d'échange collaboratif, à savoir, le demandeur d'accès principal (acteur), le fournisseur de services (**FS**) et le fournisseur d'identité (**FI**). Le rôle du STS est de gérer l'interopérabilité entre ces différentes parties dans le cas où elles adoptent des politiques différentes. Le STS se charge ainsi de fournir au FS un jeton certifiant l'identité de l'acteur authentifié par un FI, comme c'est le cas dans SAML. Plus précisément, le STS convertit localement les jetons issus de la part d'un FI au format supporté par les fournisseurs de services (ciblés).

Le service WS-Federation est une extension de WS-Trust [3] dans le sens où il se base sur la facilité d'échange des jetons de sécurité assuré par les STSs afin d'établir un contexte de confiance entre des environnements hétérogènes. En effet WS-Federation est un cadre plus global de la notion de fédération que WS-Trust. La valeur ajoutée par WS-Federation est qu'à travers ses extensions de l'ensemble des protocoles WS-Trust, il permet à ce dernier d'intégrer des attributs supplémentaires (*e.g.* un pseudonyme) aux jetons STSs. Cela peut être utile quand il est utilisé avec des *claims-authorization services* [32] dans la perspective de préserver la confidentialité (privacy) des acteurs à travers les frontières des organisations fédérées. Par exemple, dans le cas où le fournisseur de services réclame (à travers des politiques de contrôle d'accès) certains attributs d'identité d'un acteur nécessaires (pour son autorisation), le fournisseur d'identité peut vérifier que le profil de l'acteur en question est adéquat (dispose des attributs) et certifier cela au FS sans lui divulguer l'identité de l'acteur en question.

32. L'autorisation Claims-based est une approche dans laquelle les décisions d'autorisation d'accès se font par rapport à une logique arbitraire d'autorisation qui se base sur des données contenues dans des claims (packages qui contiennent des informations sur le profil d'un acteur).

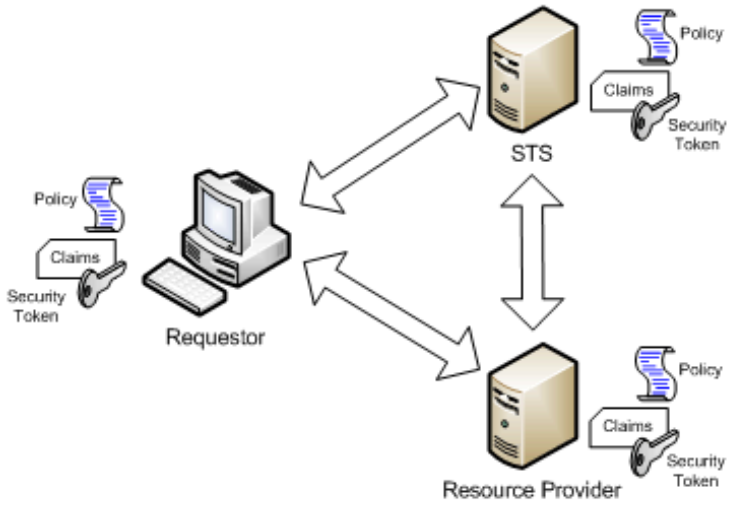

FIGURE 3.5 – WS-Trust [4].

Discussion

Nous avons besoin d'une approche de fédération flexible dans le sens où elle sera plus ouverte à la diversité en matière de protocole d'authentification. En effet, nous considérons que dans un environnement ouvert tel qu'un RSE, la possibilité qu'une entreprise puisse préserver son mécanisme d'authentification encourage et facilite la collaboration interentreprises. Cependant, le problème d'hétérogénéité entre les mécanismes d'authentification des entreprises sera toujours présent. Par conséquent, nous devons trouver une alternative à un système de fédération commun basé sur un mécanisme de délégation de certification et redirection entre plusieurs parties (STS). Par ailleurs, il est important de réduire la charge d'administration du fournisseur d'identité, de telle sorte qu'il ne soit pas sollicité pour chaque requête de demande d'accès afin de certifier l'identité du sujet de cette dernière. Dans la section suivante, nous allons nous tourner vers le contrôle d'accès dans les environnements collaboratifs.

3.2 Contrôle d'accès

Dans la littérature, de nombreux travaux dans le domaine du contrôle d'accès existent afin de répondre aux différents besoins rencontrés au fil des années avec l'évolution des outils informatiques et leurs utilisations. L'idée principale dans la conception d'un mécanisme de contrôle d'accès est de spécifier un ensemble de règles à travers un schéma d'autorisation. Ces règles indiquent quelles **actions** sont autorisées (ou non-autorisées) vis-à-vis d'un **acteur** sur de(s) **ressource**(s), selon un ensemble préétabli d'objectifs de sécurité [170].

Dans un système d'information, le noyau d'une règle de contrôle d'accès est composé de trois entités, principalement un **sujet** (l'entité active), et un **objet** (l'entité passive). Ces deux entités sont accompagnées d'une *action* qui indique le but de la requête du sujet auprès de la ressource en ques-

tion. Ensemble, les trois composants le *sujet*, l'*objet* et l'*action* sont appelés la **cible** d'une règle de contrôle d'accès.

Après une procédure d'authentification d'un sujet, et la réception des informations concernant la cible de sa requête, le composant de contrôle d'accès accorde ou refuse la demande en se basant sur les informations fournies et l'ensemble des règles de contrôle d'accès reliées au sujet ou à la cible de la requête. Un ensemble de règles ayant un critère commun (*e.g.* le sujet) est appelé **politique** de contrôle d'accès.

Dans cette section, nous allons en premier introduire deux paradigmes de modélisation d'une politique de contrôle d'accès, ainsi que les principaux composants nécessaires à leur construction. Nous allons ensuite étudier les principaux modèles de politiques de contrôle d'accès.

3.2.1 Paradigmes de construction d'un langage de politique

Une politique de contrôle d'accès est un ensemble de règles qui déterminent les droits d'accès vis-à-vis d'un utilisateur authentifié. Une politique peut par ailleurs être éventuellement soumise à certaines conditions par rapport à un certain objectif défini par le contexte. Dans la littérature, les règles de contrôle d'accès sont généralement définies sur la base de deux modèles, à savoir *Événement/ Condition/ Action*, ou, *Condition/ Action* [91].

Le modèle *Condition/ Action* est basé sur le paradigme suivant, ***Si*** *(Condition),* ***Alors,*** *(Action)*. Ce qui signifie que l'occurrence d'une action est conditionnée par une ou plusieurs contraintes. Si la condition est satisfaite, alors l'action peut se produire. Quant au premier modèle *Événement/ Condition/ Action* (*ECA*), l'action est déclenchée par l'occurrence d'un événement prédéfini. En d'autres termes, quand un événement se produit, les contraintes qui conditionnent la validation de l'action reliée sont évaluées par le mécanisme de contrôle d'accès.

Contrairement au paradigme CA, le paradigme ECA est capable de considérer le contexte en temps réel dans le processus de contrôle d'accès vu qu'il est basé sur des événements inférés à partir des attributs de l'environnement. Cela signifie que, grâce au paradigme ECA, les changements du contexte peuvent être pris en compte dans les prises de décisions. Le paradigme ECA a été adopté dans de nombreux langages de politiques comme *PDL* [131] et *Ponder* [63].

Afin d'adapter notre modèle de politiques aux différentes caractéristiques des RSE, à savoir le contexte dynamique, la co-propriété de ressources, la consistance de la gestion des politiques centrée sur l'utilisateur, et l'aspect de partage temporaire, nous avons étudié les concepts suivants :

- **Permission** : la règle qui autorise l'accès vis-à-vis d'un acteur sur une cible précise. Dans un environnement collaboratif ouvert, il est plus judicieux que l'accès à toute ressource partagée soit par défaut interdit, et que les permissions soient les exceptions qui définissent les autorisations d'accès aux ressources.
- **Interdiction** : en logique déontique une interdiction est le contraire d'une permission. Une règle de prohibition [98, 28] interdit l'accès à un acteur donné vis-à-vis d'une ressource donnée. Dans le contexte RSE, une interdiction peut servir à suspendre une autorisation d'accès définit dans le système, ou la revendiquer dans le cas de multiples propriétaires.

- **Événement** : capté par un système de supervision (monitoring) en temps réel, un événement représente tout ce qui se produit et qui est susceptible de changer, directement ou indirectement, l'état de l'environnement du système. Par exemple, l'arrivée d'un nouvel utilisateur ou une nouvelle ressource, la réception d'une requête de demande d'accès. Ainsi, le principal avantage derrière la considération des événements est la possibilité de prise en considération des changements dynamiques en temps réel dans le contexte de collaboration.
- **Condition** : Une condition est la contrainte sous laquelle une permission est approuvée. Pratiquement, une condition est un prédicat qui peut être évalué à *vrai*, *faux* ou *pas applicable* [91]. Une condition est souvent relative au contexte, *e.g.* le temps, la position géographique, le type d'application, *etc*.
- **Abstraction** : l'abstraction d'une partie de la cible (ou toute la cible) composant une politique consiste à la définir d'une manière plus généralisée en se basant sur un critère commun, comme c'est le cas pour le *"rôle"* qui est l'abstraction du *"sujet"* d'une cible dans RBAC ou les vues (ensemble de ressources) dans Or-BAC.
- **Formalisation** : Ce critère indique si le langage adopte une formalisation logique (de premier ordre en général) ou structurée (sous forme XML en général). Cette propriété d'une politique a un impact direct sur la capacité de vérification automatique de la cohérence des règles de la politique.
- **Vérification** : Cette propriété concerne les modèles ayant un formalisme logique, *e.g. Event-B* [13], *Alloy* [97], *Prolog* [55], *Situation-calculus* [124], *Event-Calculus* [147]. La vérification peut être effectuée grâce à des outils de raisonnement, ou à la main.
- **Délégation** : signifie que le partage des droits sur les ressources est basé sur une période limitée dans le temps.
- **Temps** : revient à la capacité du langage de politiques à gérer le temps dans la mise en application des règles d'autorisation.

Nous considérons ces concepts comme étant primordiaux pour notre modèle de contrôle d'accès destiné aux plate-formes RSEs. Dans ce qui suit, nous allons étudier l'état de l'art des politiques de contrôle d'accès comment ils répondent à ces besoins.

3.2.2 Politiques de contrôle d'accès classiques et modèles associés

Depuis le modèle de la matrice d'accès de *Lampson* proposé vers la fin des années 60, plusieurs modèles de contrôle d'accès ont vu le jour et on distingue [101, 37] : les politiques discrétionnaires (DAC Discretionary Access Control) [173], les politiques obligatoires (MAC mandatory access control) [171], ou encore les politiques à base de rôle (RBAC Role Based Access Control) [172, 99] qui sont mieux adaptés aux organisations professionnelles. En fonction des informations requises pour la prise de décision, une politique d'autorisation est mise en œuvre sous la forme d'un modèle de sécurité. En effet, un modèle de sécurité est un formalisme selon lequel les politiques de sécurité sont représentées (compréhension), vérifiées (consistance) et appliquées (exécution) pour répondre aux objectifs de sécurité du système [71]. Différents modèles de politiques ont été proposés. Les

modèles à description formelle comme *Lampson* [119], *HRU* [93] et *Take-Grant* [102] sont plutôt des modèles génériques, *i.e.* pouvant s'appliquer à toutes sortes de politiques. En outre, nous avons quelques modèles spécifiques, tels que les modèles de treillis [30, 25], muraille de chine [43], à base de rôle/organisation/équipe [169, 61, 18].

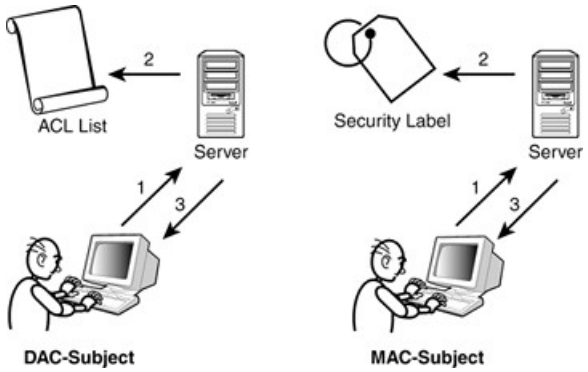

FIGURE 3.6 – MAC et DAC vue globale [89].

3.2.2.1 DAC

Une politique d'autorisation sur une ressource est dite discrétionnaire dans le cas où elle est complètement gérée par l'utilisateur qui crée cette ressource. Ce dernier est donc capable de transmettre librement les droits d'accès de la ressource qu'il détient à n'importe quel autre sujet. De plus, il est le seul à pouvoir détruire cette ressource.

DAC a été largement utilisé dans le milieu industriel et commercial [101]. Cependant, DAC souffre du problème de fuite d'informations. Cette limite est due au fait que le modèle DAC n'impose aucune restriction concernant la copie des objets. En d'autres termes, on ne peut pas empêcher un sujet ayant le droit d'accès à une ressource d'en faire une copie, et par conséquent, la partager (en tant que propriétaire) avec d'autres sujets. Cela signifie qu'au sein d'un environnement gouverné par une politique discrétionnaire, il faut faire confiance à tous les sujets qui s'échangent des informations. Ceci est loin d'être évident notamment avec les vulnérabilités des outils informatiques utilisés le plus souvent par des utilisateurs non expérimentés. Un exemple simple est le *cheval de Troie* [71]. C'est pour remédier à cette limite de confidentialité que le modèle de contrôle d'accès obligatoire a été conçu. En plus du problème de fuite d'information, un autre problème avec l'utilisation des politiques discrétionnaire est dû à leur non-flexibilité, ce qui complique davantage leur administration. En effet, pour l'ajout de toute entité active dans le système, les règles doivent être redéfinies [71].

3.2.2.2 MAC

Contrairement au système discrétionnaire, les politiques obligatoires (ou mandataires / multi-niveaux/ de treillis) sont définies non pas par le propriétaire de la ressource, mais par l'administrateur du système dans lequel les ressources sont partagées. Plus précisément, les politiques MAC se basent

sur des étiquettes de classification du niveau de sensibilité de la ressource et du niveau d'intégrité du sujet.

Une étiquette de sécurité est affectée en tant que niveau d'habilitation à chaque sujet et en guise de classification de la sensibilité pour chaque objet. Ainsi, les permissions sont basées sur une relation de comparaison entre le niveau de sensibilité de la ressource demandée avec le niveau de fiabilité du sujet en question. Les modèles de politiques obligatoires sont composés de deux grandes catégories, ceux orientés confidentialité, *Bell-Lapabula* [25] et les murailles de Chine [30] et ceux orientés intégrité, *Biba* [30].

Dans les politiques de **confidentialité** du type *Bell-LaPadula*, le contrôle d'accès se fait sur la base de deux propriétés :

- la *propriété simple* qui stipule qu'un sujet ne peut pas lire une information labellisée d'un niveau supérieur au sien.
- la *propriété étoile* interdit à un sujet de modifier (écrire) dans une ressource de plus bas niveau.

Par ailleurs, on parle aussi [19] de propriété de *tranquility* qui peut être forte ou faible pour, respectivement, permettre la mise à jour des étiquettes de sécurité en cours d'exécution du système de sécurité, ou pas. Cette propriété, connue aussi sous *"high water mark principle"*, initialise le niveau de sensibilité d'une ressource dans une session au plus bas niveau, et le met à jour au fur et à mesure en fonction des besoins de sécurité.

Quant au modèle de *Muraille de Chine*, sa principale vocation est la séparation des entités collaboratives par le cloisonnement de cercles d'échanges d'informations sur la base des conflits d'intérêts. En effet, les ressources appartenant à des entreprises concurrentes et ne pouvant pas être accédées par le même utilisateur sont enregistrées dans une *classe de conflit d'intérêt* (*CIC*[33]) . En d'autres termes, quand un utilisateur accède à une ressource$_X$ appartenant à une entreprise donnée et qu'il tente d'accéder ensuite à d'autres ressources qui font partie de la même classe CIS, l'accès lui sera refusé, et ce même s'il avait initialement le droit d'accéder à ces ressources.

Concernant le modèle d'*intégrité* de *Biba*, il est assez similaire à celui de *Bell-LaPadula* sur le plan conceptuel. La principale différence entre les deux modèles est que le modèle de *Biba* met davantage l'accent sur l'aspect d'intégrité de ressource que le modèle *Bell-Lapabula*. Pour mettre l'accent sur l'intégrité, ce mécanisme fonctionne suivant le schéma inverse que celui de *Bell-LaPadula*. Ainsi, un sujet peut lire des objets d'un niveau supérieur, mais ne peut écrire sur des objets de niveaux inférieurs.

Ces modèles multi-niveaux *MAC* ont montré leur applicabilité notamment dans les domaines restreints et fédérés par la confiance comme l'armée et le renseignement, et ce grâce à l'immunité qu'ils assurent contre les *chevaux de Troie*. Cependant, dans l'environnement ouvert du RSE, les modèles de politiques obligatoires souffrent de la limite de dégradation des niveaux d'intégrité [71]. En effet, suite à une dégradation de niveau de confidentialité (modèle Biba), ou à sa hausse qui peut parfois converger vers le niveau de confidentialité maximal (modèle Bell-LaPabula), un problème de classification est clairement identifié. En outre, les modèles multi-niveaux nécessitent une administration unique et centralisée, ce qui ne convient pas vraiment à la nature distribuée des communautés RSE où

33. conflict of interest class.

plusieurs entreprises indépendantes collaborent. Une autre famille des modèles de contrôle d'accès est basée sur le rôle des sujets des requêtes de demande d'accès, à savoir la famille *"RBAC"*.

3.2.2.3 Politiques de contrôle d'accès basées sur le rôle –RBAC-

Dans les politiques RBAC, le rôle est une façon de modéliser un ensemble de fonctions à la charge d'une entité active au sein d'une organisation. Chacune des fonctions définissant le rôle correspond à un ensemble de tâches qui donnent droit à certains privilèges au sein de l'organisation. En d'autres termes, un rôle est à l'égard d'un utilisateur au sein d'une entreprise, l'abstraction d'un ensemble de permissions d'accès. Par définition, un rôle est associé à un certain nombre de droits d'accès aux ressources protégées. Ainsi, dans une entreprise où les rôles sont prédéfinis, la définition des permissions à l'égard d'un acteur donné se fait par l'affectation de ce dernier à un ou plusieurs rôles (figure. 3.7), lui donnant accès aux droits associés aux rôles en question.

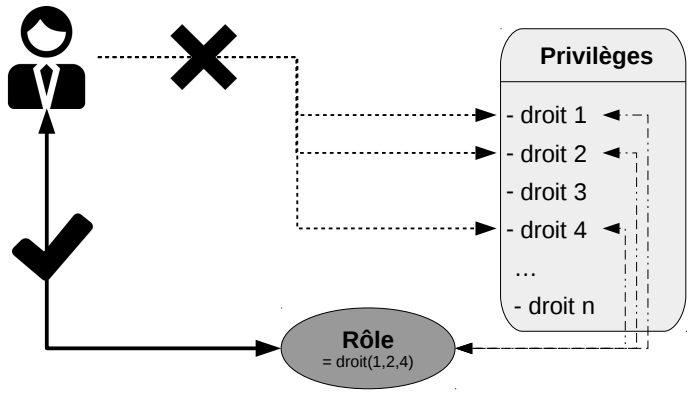

FIGURE 3.7 – RBAC vue globale.

Le modèle initial de RBAC *RBAC96* a été proposé par *Sandhu et al.*, dans [172]. Étant initialement composé de : utilisateur, permission, sessions, attribution et activation de rôle, *RBAC0* a été en premier lieu amélioré pour supporter la hiérarchie des rôles (dans *RBAC1*). Ensuite, des contraintes d'activation de rôle ont été rajoutées dans la version RBAC2 [75], (figure. 3.8).

Les rôles peuvent être structurés selon une hiérarchie qui reflète une ligne d'autorité multi-niveaux entre certains rôles dans l'entreprise. Cela se fait par l'héritage des permissions tout en évitant les conflits entre les rôles. Par exemple, dans un laboratoire de recherche, le rôle *doctorant* peut hériter les permissions du rôle *membre_du_laboratoire*. La gestion des conflits (statique ou dynamique) se fait grâce à la *séparation des devoirs*.

Dans RBAC, il existe en effet la notion de session grâce à laquelle les rôles d'un utilisateur sont activés, ou désactivés en fonction de contraintes dites de séparation des devoirs. Le rôle activé dé-

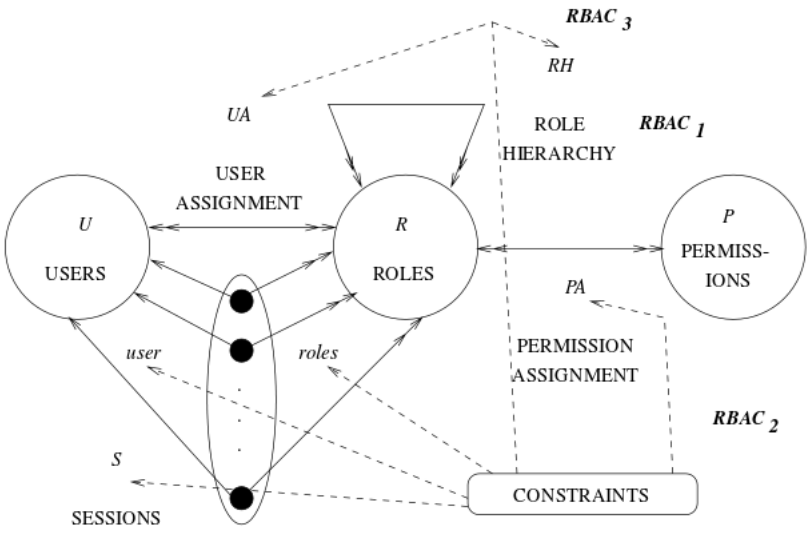

FIGURE 3.8 – Modèles RBAC [157].

termine les autorisations qui sont disponibles pour l'utilisateur à un moment donné au cours de la session. Ceci peut être utile dans la perspective de séparer les devoirs d'une manière dynamique. Le principe de séparation des devoirs vise à réduire le risque de compromettre le système de sécurité par un utilisateur en limitant ses droits légitimes par le moyen de contraintes supplémentaires [27, 16, 186]. Dans cette politique obligatoire (*"separation of duties"*) introduite par *Clark* et *Wilson* [54], deux fonctions (ou plus) affectées au même utilisateur peuvent être complémentaires mais pas être utilisées en même temps. Par conséquent, si l'utilisateur veut utiliser une permission liée à un des deux rôles, il devra désactiver l'autre(s) rôle(s). De plus, afin de garantir l'intégrité des ressources, l'enjeu principal dans l'attribution des rôles dans RBAC, est d'associer les droits minimums à un utilisateur pour la réalisation d'une opération donnée, *i.e. "least privilege"* à travers le rôle correspondant.

RBAC a été initialement conçu pour gérer le contrôle d'accès (en optimisant l'administration des politiques) au sein d'une même entreprise. Une entreprise composée de plusieurs personnes peut être considérée comme un environnement collaboratif. Cependant, cet environnement collaboratif interne reste restreint dans le sens où le mode de fonctionnement de l'entreprise, avec les tâches affectées à chaque membre de l'entreprise sont connus à l'avance. RBAC a également été utilisé dans des environnements plus ouverts avec des structures collaboratives davantage hétérogènes, tels que les équipes et les milieux multi-organisationnel.

RBAC Extensions collaboratives

De nombreux travaux inspirés du modèle RBAC se sont orientés vers des environnements collaboratifs comme les équipes, les organisations et les réseaux sociaux. On trouve, par exemple *Organization Based Access Control OrBAC* [111], *Multi-organization Based Access (control Multi-OrBAC)* [72],

Team based access control [185, 195, 83], *Relationship-based access control* [53].

Dans le modèle des politiques basées sur la notion d'équipe *TMAC*, l'entité principale est l'*"équipe"* qui consiste en une abstraction d'un ensemble de sujets dans une équipe de collaboration ayant un objectif commun. La spécificité dans ce modèle est que l'attribution des rôles et l'activation des permissions sont gérées séparément. Plus précisément, un utilisateur obtient par le biais de son appartenance à une équipe le droit d'accès aux ressources de l'équipe. Cependant, le droit d'accès octroyé à l'utilisateur dépendra de son rôle ainsi que l'activité courante de l'équipe en question. Par exemple [84], un médecin est habilité à prescrire des traitements, en revanche, il se peut que la politique de l'hôpital où il exerce lui permette de ne traiter uniquement que les patients dont il dispose d'un historique de suivi. Grâce à *TMAC*, cette politique peut être mise en œuvre. Il suffit d'assigner le médecin à l'équipe de traitement du patient, ainsi il sera en mesure d'accéder aux informations de suivi du patient. À ce dernier niveau, le rôle du médecin peut être utilisé pour définir le niveau de granularité de l'accès aux informations du patient.

Quant aux modèles *OrBAC* et *Multi-OrBAC*, ils sont basés sur une conception qui va au-delà de l'abstraction du sujet par un rôle ou une équipe. En effet, dans *OrBAC* les éléments de la cible d'une règle, à savoir le *sujet*, la *ressource* et l'*action*, sont respectivement abstraits par le *rôle*, la *vue* et l'*activité*. *OrBAC* permet de définir plusieurs types de règle de contrôle d'accès, à savoir *des permissions, des interdictions, des obligations* ainsi que *des délégation*. En outre, *OrBAC* supporte la modélisation du contexte dans les règles de contrôle d'accès. Par ailleurs, une règle dans *OrBAC* peut avoir plusieurs types, à savoir *"Permission"*, *"Interdiction"* ou *"Obligation"*. Ainsi, une règle dans *OrBAC* prend la forme du quintuplet : *Type-de-règle (org, rôle, activité, vue, contexte)*. La délégation est gérée par un ensemble de prédicats basés sur le type *Permission* [26]. Le schéma de vérification d'une règle est le suivant : dans l'organisation *org*, si le *sujet* possède le bon *rôle*, et l'*action* désirée appartient à l'*activité*, la *ressource* demandée fait partie de la *vue*, et le *contexte* est validé entre le *sujet*, l'*action* et la *ressource*, alors le sujet peut obtenir l'accès demandé. La variante *Multi-OrBAC* [71] est une adaptation du modèle *OrBAC* initial pour les environnements multi-organisationnel distribués et hétérogènes. La nouveauté est que l'abstraction des *rôles*, *activités* et *vues* devient liée à l'organisation, *i.e. rôle/activité/ressource_dans_Organisation*. Ainsi, le modèle d'une règle d'un sujet appartenant à l'organisation Org_1 souhaitant accéder à une ressource appartenant à l'organisation Org_2 devient : *Permission/Prohibition/Obligation (Rôle dans* Org_1*, Activité dans* Org_2*, Vue dans* Org_2*, Contexte dans* Org_2*)* [34]. L'implantation de ce modèle est basée sur une administration centralisée qui favorise l'interopérabilité en matière de gestion des accréditations dans les environnements collaboratifs.

3.2.2.4 Synthèse

Dans un environnement collaboratif social, le partage de ressource est généralement centré sur l'acteur collaboratif (*user-centric*) [136, 88, 92]. Le contrôle d'accès discrétionnaire DAC [173] répond à cette caractéristique vu que le partage de ressources se fait à la discrétion de l'acteur qui détient la ressource. En revanche, les politiques DAC sont difficiles à administrer car il faut les remettre à jour

34. L'utilisation des majuscule pour chaque composant de la *cible* signifie qu'il s'agit d'instances plutôt que variables.

à chaque fois qu'un nouvel utilisateur rejoint une communauté de collaboration donnée. De plus, le contrôle discrétionnaire est sensible à la fuite d'information et donne un pouvoir exagéré aux acteurs dont certains peuvent être malintentionnés et nuire ainsi à la sécurité du système.

Donc, un besoin de contrôle sur les politiques des acteurs collaboratifs s'impose. Cela peut être résolu par le contrôle d'accès obligatoire MAC qui se base sur des contraintes reliées au niveau d'habilitation du sujet ainsi que la classification de la sensibilité de la ressource. Néanmoins, ces contraintes fortes pour les entreprises [71] sont difficiles à gérer dans la pratique, notamment dans un cadre collaboratif multi-organisationnel. En effet, s'il est possible d'apporter une évaluation pertinente sur l'intégrité de ressources au sein d'une même entreprise, cette tâche reste beaucoup plus compliquée (et souvent source de désaccords) quand il s'agit de plusieurs entreprises indépendantes, éventuellement concurrentes. De plus, le niveau d'intégrité dans les modèles MAC converge facilement vers le niveau extrême [35], et par conséquent bride la collaboration. Quant à l'administration des politiques, elle n'est pas meilleure par rapport au contexte RSE que celle des politiques DAC. En effet, les politiques MAC nécessitent des mises à jour souvent manuelles des labels de classifications des ressources et/ou acteurs.

Pour faire face à ce besoin d'adaptation aux changements fréquents en matière d'acteurs dans l'environnement RSE, le modèle RBAC [172] se présente comme une meilleure alternative en réduisant considérablement cette complexité de mise à jour de politique. En effet, RBAC a été très attractif dès ses premières applications, car il est possible de le configurer de telle sorte qu'il supporte les politiques MAC et DAC [76]. Dans une politique RBAC, l'idée principale consiste à regrouper plusieurs acteurs selon un même critère d'habilitation, de statut ou de compétences pour la réalisation d'un certain nombre de tâches. Ainsi, l'administration des politiques RBAC se focalise en majeure partie sur la gestion des rôles. Par exemple, pour chaque nouvel acteur qui rejoint une communauté, un ancien et/ou nouveau rôle est attribué. Cependant, le modèle RBAC initial a été victime de son propre succès, car il n'est en effet pas vraiment adapté à de nombreux contextes collaboratifs dans lesquels on a essayé de l'utiliser [101]. La principale limite du modèle RBAC est le manque de flexibilité dans la définition des contraintes sur les rôles de telle sorte que pour chaque restriction, un nouveau rôle doit être défini.

Au fur et à mesure de l'utilisation du modèle RBAC dans divers domaines, de nouveaux besoins ont été relevés et ont donné lieu à de nombreuses extensions, comme *TMAC*, *OrBAC* et *multi-OrBAC* [185, 111, 72]. Néanmoins, plusieurs limites liées à l'utilisation des politiques RBAC et ses extensions persistent dans un environnement RSE. D'abord, la question se pose par rapport au défi de n'attribuer que le *moindre-privilège* à un utilisateur donné dans la procédure d'affectation de rôle. En outre, la gestion des rôles reste une tâche non évidente, car il faut une bonne connaissance du mode de fonctionnement des entreprises avec les besoins nécessaires en matière de personnel par rapport aux objectifs de chaque entreprise. Il a même été démontré par *Jianfeng Lu et al.* dans [133] que le problème de mise à jour des rôles est, dans certain cas, linéaire. Néanmoins dans le cas général il s'agit d'un problème de complexité d'ordre *NP-complet*. Par ailleurs, à l'image de l'hétérogénéité de l'environnement RSE, les différents modèles d'extensions de RBAC tels que TMAC, Or-BAC et Multi-

35. Maximal ou minimal en fonction du modèle intégrité/confidentialité.

OrBAC rajoutent de la complexité dans la définition des politiques de contrôle d'accès [182]. Cela est dû à l'hétérogénéité liée à la manière dont les entreprises définissent les rôles et les privilèges qui leurs sont associés. De plus, ces modèles manquent de flexibilité dans le cas de reconstruction d'équipe ou de réorganisation hiérarchique d'une organisation.

Pour résumer, les modèles *X-RBAC* ne peuvent pas être "la solution" pour notre contexte RSE. Car, outre le phénomène d'explosion de rôle et le fait que les gens changent assez fréquemment leurs travail dans le milieu professionnel, la collaboration peut être compliquée dans le cas de différentes sémantiques d'un même intitulé de rôle dans deux entreprises indépendantes.

Ainsi, dans le but d'obtenir un cadre de collaboration fluide sur le long terme, la notion du rôle doit être flexible de telle sorte qu'elle puisse facilement homogénéiser les sémantiques des rôles entre les différents partenaires. Cela est possible avec la combinaison du rôle avec d'autres attributs de profil de collaboration, *e.g.* la réputation, l'expérience professionnelle, le niveau d'habilitation, la position géographique, *etc*.

3.2.3 Attribute Based Access Control

Selon Vincent Hu et ses collègues au NIST[36], *"Attribute-based access control (ABAC) is a flexible approach that can implement AC policies limited only by the computational language and the richness of the available attributes, making it ideal for many distributed or rapidly changing environments"*[95]. Nous nous sommes en premier lieu inspirés de cette définition pour étudier les avantages de la modélisation *"ABAC"* dans notre contexte RSE.

En effet, vu que l'identité du demandeur, son rôle ou son groupe (organisation) ne sont pas assez suffisants pour exprimer des politiques dans un monde multi-organisationnel hétérogène [95], l'idéal est d'avoir plus de liberté et de choix pour la définition d'une politique. Cette motivation a été l'idée principale derrière la conception des politiques ABAC. Comme le montre la figure 3.9 une politique ABAC se base sur un mix entre certains attributs choisis d'une manière arbitraire concernant : le demandeur d'accès (le sujet), l'objet désiré (la ressource) ainsi que les attributs de l'environnement (le contexte). Un attribut prend la forme d'une paire **Nom : Valeur**.

Par ailleurs, la modélisation ABAC permet de garder les avantages des politiques classiques telles-que *DAC, MAC, RBAC ou Or-BAC*, tout en remédiant à leurs limites respectives. En effet, ABAC offre plus d'expressivité à l'égard des composants les plus importants pour la définition d'une politique de contrôle d'accès, à savoir, le sujet, l'objet et le contexte. Par exemple, l'abstraction du sujet via un rôle, équipe ou une organisation peut être un simple attribut qui s'ajoute à ceux qui définissent le profil du sujet en question. La flexibilité d'ABAC concerne également le modèle MAC (*i.e.* les niveaux d'intégrité) et le modèle DAC (*i.e.* les listes des identités des acteurs autorisés). Dans [101], on trouve une démonstration d'équivalence entre ABAC et les modèles classiques de contrôle d'accès. De plus, la modélisation du contexte devient très flexible car différents attributs comme le temps, l'adresse IP, le type de terminal, *etc.* peuvent être facilement modélisés (*i.e.* sous la forme **Nom : Valeur**) et intégrés dans les règles et les politiques de contrôle d'accès. ABAC permet, en outre, de réunir la

36. National Institute of Standards and Technology

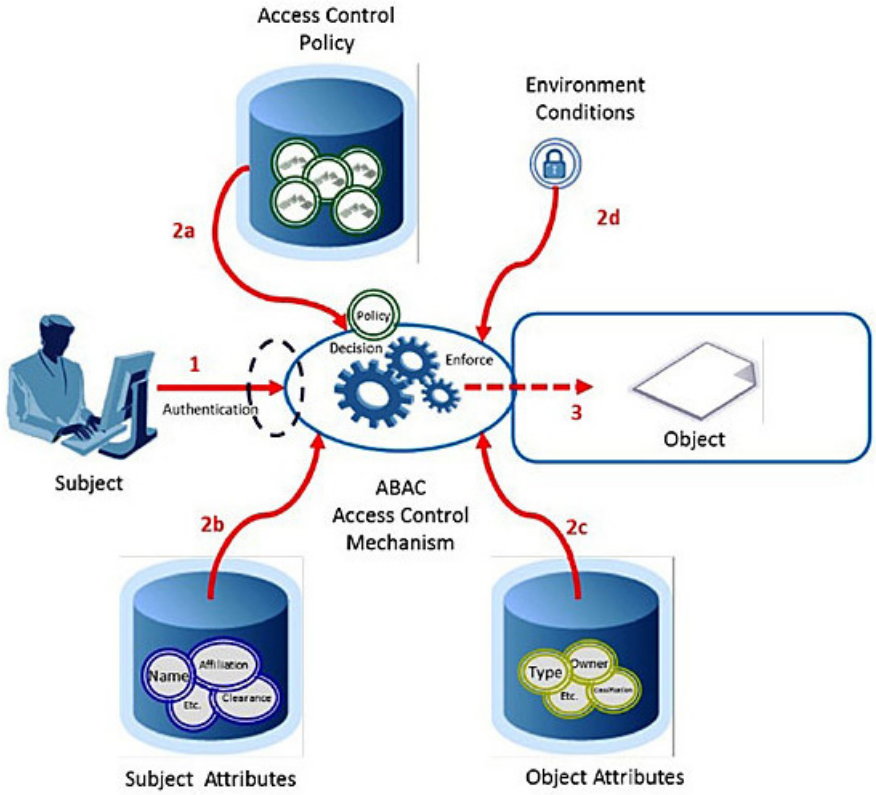

FIGURE 3.9 – Scénario de contrôle d'accès ABAC [94].

plupart des extensions de RBAC sous un même framework ABAC [101].

En outre, grâce à ABAC nous avons une administration moins coûteuse des règles. En effet, afin de modéliser toutes les règles possibles à partir d'un ensemble d'attributs binaires (*i.e.* pas de répétition du même attribut dans la même règle) ABAC nécessite 2^n règles pour n attributs dans le pire des cas, comparée à 2^n rôles dans RBAC [117] par exemple.

Nous croyons vivement à la compatibilité et l'adaptabilité du modèle ABAC à notre contexte des RSEs. En effet, grâce aux attributs qui modélisent les règles de contrôle d'accès, ABAC nous permet de répondre à plusieurs besoins, notamment l'abstraction des cibles de règles, la flexibilité de définition de conditions et obligations, les politiques d'interdictions et la flexibilité dans la gestion des contraintes du contexte.

Par rapport à la logique déontique [108, 100, 60, 63] qui permet de définir des règles d'interdiction, ABAC assure la facilité de changement de l'état d'une règle en basculant son type d'une permission à une interdiction, et ce, simplement grâce à la mise à jour de l'attribut qui désigne le type de la règle. Par ailleurs, une ressource peut être la copropriété de plusieurs acteurs [88] faisant

partie de la même entreprise ou d'entreprises distinctes. Cela qui signifie qu'il faut, en outre, prendre en compte l'aspect combinaison de règles et politiques de contrôle d'accès [142, 98, 145, 99, 63]. Cela peut être bien traité grâce aux stratégies de combinaison de règles dans des ensembles de politiques et des *policySet*s de XACML (que nous aborderons dans ce qui suit).

3.2.4 Mise en œuvre de politiques ABAC

Dans cette section nous allons aborder l'aspect de la mise en œuvre de politique de contrôle d'accès. Nous allons nous focaliser sur la mise en œuvre d'un modèle ABAC.

3.2.4.1 eXtensible Access Control Markup Language "XACML" - un formalisme structuré -

XACML *"eXtensible Access Control Markup Language"* est un standard d'*OASIS* fortement lié à l'implantation des politiques ABAC. XACML [164] fournit deux facettes dans le domaine du contrôle d'accès, à savoir, un langage de définition de politiques, et aussi un language de décision (*i.e.* requêtes et/ou réponses) pour le processus du contrôle d'accès. Le langage de politiques est utilisé pour décrire les exigences générales pour le contrôle d'accès et sert à définir de nouvelles fonctions, types de données, logique de combinaison, *etc.*. Quant au langage de requête/réponse, il permet de formuler une requête de demande d'accès afin de savoir si oui ou non une action donnée doit être autorisée, et interprète le résultat de cette requête. Le résultat de cette requête inclut toujours une réponse qui permet de savoir si la demande devrait être autorisée. Le format de la réponse prend une des quatre valeurs suivantes : *Permit, Deny*, *Indeterminate* (au cas où de manque de paramètres et/ou erreur) [37] ou *Not Applicable* (le service en question ne peut pas répondre à la requête).

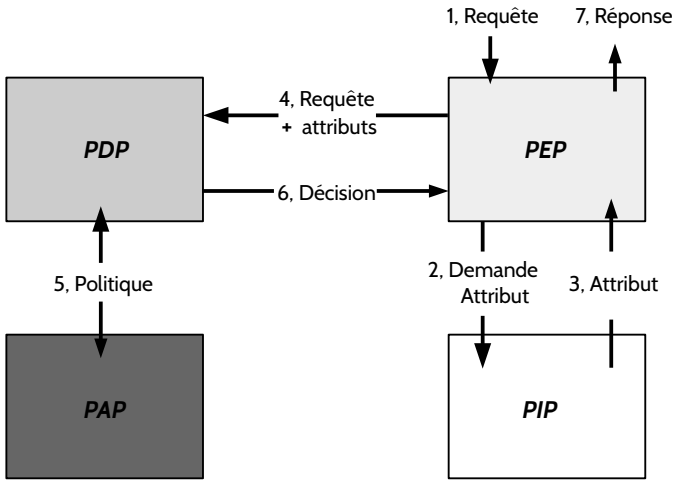

FIGURE 3.10 – XACML vue abstraite.

37. Si une erreur est survenue ou une valeur requise manquait, et par conséquent la décision ne peut être calculée.

Lorsqu'un acteur veut accéder à une ressource, une requête est établie auprès de l'entité qui protège la ressource en question, à savoir le *Policy Enforcement Point* (PEP). Le PEP forme de son côté une requête basée sur les attributs du demandeur, la ressource en question, l'action, et d'autres informations relatives à la demande. Le PEP envoie ensuite cette demande à un autre point chargé de la prise de décision, à savoir le *Policy Decision Point (PDP)*. Afin de produire sa réponse concernant l'approbation de l'action, le PDP examine la requête par rapport à un ensemble de politiques. Ensuite, la réponse du PDP est envoyée au PEP, qui peut alors autoriser ou refuser l'accès au demandeur. Le PEP et le PDP peuvent aussi bien être contenus dans une seule application, ou bien être répartis (*i.e.* distribués) sur plusieurs serveurs, comme la plate-forme Cardea [123]. La figure 3.10 illustre les composants ainsi que le mode de fonctionnement de XACML. Il existe aussi deux modules complémentaires pour l'accomplissement du processus de contrôle d'accès, à savoir :

- le **PIP** (Policy Information Point) l'entité où les informations (attributs) relatives aux utilisateurs, ressources et l'environnement sont stockées.
- le **PAP** (Policy Administration Point) le conteneur des règles et politiques de contrôle d'accès.

Comme illustré dans la figure 3.10, le processus se déroule ainsi :

1. le **PEP** reçoit une requête d'autorisation d'accès à une ressource,
2. le **PEP** récupère auprès du **PIP** les attributs nécessaires pour l'établissement d'une requête conforme afin de la transmettre au **PDP**,

3+4. le **PEP** transmet au **PDP** la requête bien formulée : l'environnement (contexte), le sujet (consommateur), l'action et la ressource désirée,

5. le **PDP** interagit avec le **PAP** pour récupérer les politiques d'accès précédemment établies ou établies par rapport aux nouvelles valeurs,

6+7. le **PDP** prend une décision (en fonction des attributs de la requête reçue) et la transmet au PEP qui à son tour retransmet la réponse d'autorisation d'accès à la ressource à l'utilisateur principal.

La réponse finale du PDP peut être

- une autorisation, *i.e.* **Permit**,
- un refus, *i.e.* **Deny**,
- aucune règle ne s'applique à la requête, *i.e.* **Not applicable**.
- ou bien **Indeterminate** dans le cas où des problèmes d'exécution sont détectés

Politiques de contrôle d'accès XACML

À la racine de chaque document de politique XACML, on trouve une Policy (figure 3.11) ou un PolicySet. Une Policy regroupe deux ou plusieurs règles (*Rules*). Deux ou plusieurs politiques (*Policy*) peuvent également être regroupées au sein d'un ensemble appelé *PolicySet*. Ainsi, les processus d'évaluation d'une Policy et d'un PolicySet XACML nécessitent respectivement des mécanismes de combinaison de règles et de politiques. Comme mécanisme de combinaison, nous pouvons appliquer les stratégies suivantes :

```
<Policy PolicyId="SamplePolicy"
       RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">

  <!-- This Policy only applies to requests on the SampleServer -->
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">SampleServer</AttributeValue>
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                                     AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
      </ResourceMatch>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>

  <!-- Rule to see if we should allow the Subject to login -->
  <Rule RuleId="LoginRule" Effect="Permit">

    <!-- Only use this Rule if the action is login -->
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">login</AttributeValue>
          <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                                     AttributeId="ServerAction"/>
        </ActionMatch>
      </Actions>
    </Target>

    <!-- Only allow logins from 9am to 5pm -->
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal"
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
                                        AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal"
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
                                        AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValue>
      </Apply>
    </Condition>

  </Rule>

  <!-- We could include other Rules for different actions here -->

  <!-- A final, "fall-through" Rule that always Denies -->
  <Rule RuleId="FinalRule" Effect="Deny"/>

</Policy>
```

FIGURE 3.11 – Politique XACML [164].

- **Deny-Overrides** : si au moins une règle/politique est non autorisée, la requête ne sera pas approuvée.
- **Permit-Overrides** : si au moins une règle/politique est autorisée, la requête sera approuvée.
- **All-permit** : pour qu'une requête soit approuvée, toutes les règles/politiques doivent être autorisées.
- **All-Deny** : pour qu'une requête soit non approuvée, toutes les règles/politiques ne doivent pas être autorisées.
- **All-Not-Applicable** : si toutes les règles/politiques ne sont pas applicables, faute d'un manque de paramètre(s) ou erreur(s) interne(s).

Néanmoins, afin d'éviter les problèmes d'incohérences, toutes ces stratégies de combinaison ne

peuvent pas être utilisées ensemble dans le même modèle de politique ou de PolicySet. Prenons l'exemple d'une politique qui contient des règles avec les deux effets *Permit* et *Deny* pour la même cible. Ainsi, si on opte pour une stratégie *Deny-Overrides*, il sera incohérent d'utiliser avec une stratégie *Permit-Overrides*, il faudra plutôt utiliser avec une stratégie *All-permit*, et vice-versa. Cependant, la stratégie *All-Not-Applicable* est utilisable avec toutes les stratégies [38].

Règles de contrôle d'accès XACML

Une règle est la plus petite unité de décision dans une politique de contrôle d'accès XACML. Une règle est composée d'une *cible*, une ou plusieurs *conditions* et un *effet*.

- **Cible** : Une des tâches du PDP est de trouver une politique qui s'applique à la requête reçue. Une cible est essentiellement un ensemble de conditions simplifiées concernant le *sujet*, la *ressource* et l'*action* qui doivent correspondre à ceux de la requête en question. XACML utilise des fonctions booléennes pour comparer les valeurs trouvées dans une requête à celles incluses dans les cibles des règles existantes. Si toutes les conditions d'une cible sont remplies, alors sa règle, ainsi que la politique et le PolicySet associés peuvent être évalués. La cible peut également être utilisée comme index pour regrouper et/ou rechercher des politiques et/ou PolicySet.
- **Condition et effet** : Une fois la cible trouvée, la règle correspondante peut être évaluée pour donner une suite (*i.e., effet*) positive ou négative à la requête en question. La réponse est ainsi basée sur l'évaluation des conditions de la règle grâce à une fonction booléenne. Cette évaluation peut renvoyer le rapport *Indeterminate* si des erreurs d'évaluations se produisent. Si toutes les conditions de la règle sont satisfaites, donc l'*effet* de la règle sera *Permit*, ce qui signifie que le sujet de la requête est autorisé à réaliser l'action désirée sur la ressource demandée. Dans le cas contraire, l'*effet* sera *Deny* et la requête reçue sera ainsi refusée.

3.2.4.2 Le point sur XACML par rapport à OpenPaaS RSE

XACML est sûrement l'un des langages les mieux adaptés pour l'implantation du modèle ABAC, car il a montré son efficacité dans le cadre de plusieurs projets à l'image de *Cardea* [123], epSOS [7], NHIN [8]. Cepedant, quand il s'agit d'environnement user-centric tel qu'un RSE, XACML présente quelques limites. En effet, bien qu'il existe des éditeurs de politiques comme *XACML studio* [5] et *UMU-XACML-Editor* [192], la conception des politiques XACML reste une tâche non évidente pour des utilisateurs non expérimentés. Par conséquent, cela risque d'induire le système en erreur, à cause d'éventuelles incohérences des règles définies. En d'autres termes, XACML est verbeux et manque de capacité de vérification automatique de la cohérence des politiques. C'est pour cela que de nombreux travaux offrant un formalisme logique à XACML ont été proposés.

Dans [114], les auteurs se basent sur la logique descriptive et les ontologies (OWL) pour formaliser XACML dans le but de pouvoir faire des analyses de consistance des politiques XACML, *i.e.* le raisonnement logique. Pour cela, ils se basent sur un raisonneur de Logique de Description. Dans [45],

38. De plus amples détails sur cet aspect seront présentés dans le chapitre **cf. Contrôle d'accès**

les auteurs formalisent les politiques XACML en utilisant l'algèbre de processus CSP (Communicating Sequential Processes). Ils se basent sur une comparaison de politiques en utilisant du Model-Checking pour la vérification de la consistance. Dans [138], les auteurs proposent une méthode basée sur la logique du premier ordre pour l'analyse des intéractions et la détection et la visualisation des éventuels conflits dans les politiques XACML. Leur méthode de vérification est fondée sur le langage *Alloy*. Dans [160], les auteurs proposent une algèbre pour la modélisation des politiques de contrôle d'accès en prenant en considération, les permissions ainsi que les interdictions. Les auteurs montrent également que cette algèbre est facilement transformable en langage XACML. Dans [152, 193], X-STROWL une extension généralisée de XACML est présentée. La spécificité de ce modèle est qu'il est basé sur le rôle et supporte des contraintes géo-temporelles. En effet, à partir d'une hiérarchie de rôles (définie grâce à des ontologies OWL [184]), l'idée est de déterminer les relations entre les rôles grâce à des fonctions XACML. La modélisation avec les ontologies offre la capacité de vérification automatique au modèle concernant la hiérarchie des rôles et l'héritage des permissions. Les auteurs dans [139] définissent une sémantique formelle de XACML grâce à la grammaire *BNF* [113], qui a été implantée par la suite dans un framework basé sur *Java* et de l'outil *ANTLR*. L'idée fondamentale derrière ce travail est de clarifier toutes les ambiguïtés et les aspects complexes du standard XACML.

Par ailleurs, la gestion du contexte de collaboration est limité dans XACML. En effet, bien que les requêtes XACML sont parfois considérées comme des événements, le formalisme XACML reste limité sur ce côté. En effet, XACML ne considère pas les changements dynamiques en temps réel des attributs de l'environnement. Ce challenge a été relevé dans [153], et le résultat est un langage de politique appelé *xfACL* (formalisé en *OCaml*) qui combine XACML avec RBAC. La gestion des changements dynamiques des attributs du contexte est gérée grâce à des *politiques auxiliaires*. De plus, à l'image des précédant travaux cités qui offrent un formalisme logique au Standard XACML, à notre connaissance il n'y a pas de modèle capable de gérer les changements dynamiques temporels de l'environnement tout en étant capable de faire un raisonnement efficace sur la cohérence des règles définies par des utilisateurs non expérimentés, tel que c'est le cas dans les RSEs. En outre, la gestion du temps dans ces travaux reste limitée et statique.

Pour résumer, la prise en compte du contexte dans les différents modèles de contrôle d'accès est limitée à des situations prédéfinies. Cependant, dans un environnement collaboratif riche en changements tel qu'un RSE, il est important d'aller au delà de ces situations prédéfinies et d'inclure de manière plus dynamique les propriétés liées au changements contextuels dans le processus du contrôle d'accès. En effet, l'étude de la considération du contexte a été davantage approfondie dans le cadre d'autres travaux qui ont traité un autre type de politiques de contrôle d'accès, à savoir les politiques de contrôle d'accès *"dynamiques"*. La principale idée derrière le contrôle d'accès dynamique est que les politiques, ainsi que les décisions du contrôle d'accès, soient variables dans le temps, et ce en fonction d'un certain nombre de paramètres relatifs au contexte.

3.2.5 Contrôle d'accès dynamique

Contrairement aux modèles classiques de contrôle d'accès, le contrôle d'accès dynamique va au delà de politiques prédéfinies. Il a été développé [178, 123, 187] dans la perspective de rendre les décisions de contrôle d'accès (requêtes/réponses) plus dynamiques et réactives vis-à-vis d'informations captées en **temps réel**. Ces informations peuvent être liées à différents concepts, comme le contexte, la confiance (la réputation) et le risque [194, 44].

3.2.5.1 Contexte et contrôle d'accès

Dans un environnement collaboratif hétérogène, la mobilité dans l'utilisation des terminaux informatiques est un avantage précieux. Par exemple, le fait qu'un acteur puisse rejoindre son réseau collaboratif (équipe, entreprise, *etc.*) depuis différents types de terminaux (ordinateur, smartphone, *etc.*) et depuis n'importe quel endroit géographique (une autre villes, un autre pays, *etc.*). Cependant, cette flexibilité présente certaines vulnérabilités en matière de sécurité. Cela peut être dû au manque de fiabilité des procédures d'authentification qui va lier une identité (authentifiée) à des permissions d'accès. Plus précisément, on ne peut pas être totalement sûr qu'un acteur ne se fait pas passer pour un autre (*i.e.* usurpation d'identité). En revanche, cette vulnérabilité peut être atténuée au moyen de la vérification de certaines contraintes liées au contexte. Par exemple, si on détecte une connexion depuis une adresse IP d'un pays géographiquement très éloigné de celui où se trouve l'acteur en question, l'accès peut être alors refusé.

Le contexte représente l'aspect dynamique d'une situation [182]. Une situation dynamique change à travers le temps. Ainsi, le temps est l'une des propriétés les plus importantes du contexte, ce qui a suscité l'intérêt des chercheurs dès l'émergence des plate-formes collaboratives. En effet, la notion du temps dans RBAC a été initialement introduite par *Bertino et al.,* dans *Temporal-RBAC (TRBAC)* [27], et ce dans le but de gérer les contraintes temporelles d'activation (ou désactivation) périodique des rôles et les dépendances temporelles entre les actions autorisées. Par exemple, l'utilisateur *Bob* ne peut exécuter l'$action_B$ qu'une fois l'$action_A$ est réalisée par *Alice*. *Generalized Temporal Role based Access Control (GTRBAC)* [106] est une version plus générique du modèle *TRBAC*. Dans GTRBAC, les hiérarchies de rôle ainsi que leurs dépendances ont été traitées sur les plans temporels et sémantiques.

Par ailleurs, de nombreux travaux se sont focalisés sur l'évolution des privilèges dans le temps. Par exemple, dans *CCRR Collaborative Concur Task Trees* [144], l'activation des rôles est reliée à l'état d'avancement des tâches dans le cadre d'un objectif commun. L'idée est de décomposer les tâches complexes en plusieurs sous-tâches simples et atomiques et de gérer les concurrences entre elles. Un autre exemple d'un mécanisme de contrôle d'accès orienté *workflow* est présenté dans [96] sous le nom de *SecureFlow*. Il s'agit d'un système [39] de gestion de flux opérationnels qui se base sur une hiérarchie de rôle, un regroupement des objets (sous forme d'ensemble et sous ensembles) et un intervalle de temps et pendant lequel la tâche doit être exécutée.

39. Sous forme d'un site web.

Un environnement collaboratif tel qu'un RSE a besoin d'une vue dynamique sur le contexte, dans le sens où la modélisation du contexte ne doit pas se limiter à des situations prédéfinies. En effet, à l'image de la grande fréquence d'interactions au sein d'un environnement RSE, on peut considérer ce dernier comme étant un environnement riche en *événements*. Un événement est une action réalisée par un acteur à un instant donné et captée par un système de supervision. Ainsi, dans un environnement RSE, tous les événements ne doivent pas être totalement ignorés. C'est pour cela que la plupart des approches proposées de modélisation du contexte ne s'accordent pas vraiment avec la nature dynamique du RSE.

Encore plus loin dans la modélisation du contexte

L'aspect événementiel (introduit brièvement dans la section **cf.** *3.2.1*) a été exploité et intégré dans de nombreux travaux qui font partie de la famille des modèles de politiques ECA. Parmi ces derniers on distingue *The Policy Description Language* [131, 29], *Law-Governed Interaction* [142] et *Ponder* [63]. L'un des premiers langages spécialisé dans l'administration des réseaux, *PDL* [131, 29] est un langage de politiques dans lequel les événements peuvent être simples ou composés à base de connecteurs logiques (et/ou) ou temporels. Dans *LGI* [142], les politiques d'autorisation se focalisent sur le contrôle d'accès dans les environnements collaboratifs en contrôlant le flux d'échange de messages entre des agents distribués. Dans [63], les auteurs présentent *Ponder*, un langage de politiques basées sur le rôle pour la gestion de sécurité dans les systèmes distribués. Implanté en *Java*, *Ponder* se présente comme étant un langage déclaratif et orienté-objet. Par ailleurs, dans [115] les auteurs présentent un modèle de politiques d'autorisation et d'obligation, appelé EB^3 *SEC*, basé sur les événements dans les systèmes dynamiques. Ce modèle est fondé sur une méthode formelle basée sur une algèbre de processus, permettant de spécifier des politiques de sécurité fonctionnelles dans des systèmes d'information.

Dans un langage de politiques de contrôle d'accès dynamique, l'aspect événementiel est très avantageux car il donne la possibilité de prendre en considération les événements qui se produisent dans l'environnement collaboratif (*i.e.* contexte), et ce en temps réel. Les langages de politiques basés sur le paradigme ECA précédemment cités ont été développés dans des contextes particuliers afin de répondre à des objectifs de sécurités spécifiques. En effet, ces langages ne permettent pas de couvrir tous les besoins que nous avons relevés (**cf. chapitre Problématique et motivations**), notamment en matière de flexibilité des politiques et la gestion des permissions temporaires, *i.e.* délégations. Cela signifie qu'ils ne sont pas très bien adaptés à la nature dynamique de notre contexte de problématique, à savoir les communautés RSE. Par conséquent, nous nous sommes focalisés sur la conception d'un langage de politiques de contrôle d'accès dynamique suivant le paradigme ECA sur la base du formalisme Event-Calculus (**cf. Section 3.2.7**).

Le contexte est un cadre général qui inclut plusieurs variables relatives aux entités collaboratives. Dans une vision dynamique du contrôle d'accès dans un environnement collaboratif, certaines variables sont plus importantes que d'autres, comme la *confiance numérique* que le système de sécurité ainsi que l'ensemble des entités de collaborations accordent à un sujet donné.

3.2.5.2 Confiance numérique

Dans un processus de contrôle d'accès, l'identité d'un utilisateur est généralement soumise en premier lieu à une phase d'authentification. Néanmoins, d'un côté l'authentification ne peut pas assurer le bon usage des permissions liées à l'identité du sujet en question, et d'un autre côté elle ne peut pas garantir que le sujet authentifié aura un comportement qui ne reflète aucune menace de sécurité pour le système. Cela constitue un sérieux souci, qui s'accentue lorsqu'il s'agit d'environnement RSE où l'enjeu de perte d'information est de taille. En effet, outre l'ouverture des cercles collaboratifs à de nouveaux acteurs (inconnus), même le comportement d'un acteur interne est imprévisible dans le sens où il ne peut être garanti et est susceptible de changer au cours du temps. Par conséquent, il est important de considérer cela dans le processus d'autorisation. Ce challenge peut être résumé par la prise en compte de la confiance (*trust*). La confiance peut être considérée sur la base du comportement individuel d'un acteur dans le temps, ou simplement d'une image reflétée par la communauté de laquelle est issu le sujet en question, *e.g.* entreprise, groupe. De nombreux travaux ont mixé la notion de confiance avec le contrôle d'accès, ce qui a débouché sur plusieurs modèles de contrôle d'accès [127, 14, 182, 85, 196, 68].

Les approches existantes sous le volet de la confiance numérique peuvent être considérés sous deux grandes catégories [197], à savoir, les politiques basées sur la gestion de confiance, et les systèmes d'évaluation de réputation. Dans la première catégorie, la confiance est statique et binaire, car elle est basée uniquement sur l'authenticité de l'identité de l'acteur, et ce à travers un jeton d'authentification [40], comme c'est le cas dans RT [126], PolicyMaker [36], KeyNote [34], [35], [125]. Quant à la catégorie basée sur la réputation, elle est davantage dynamique, car l'évaluation de la confiance est basée sur l'historique des interactions des utilisateurs. En d'autres termes, l'évolution de la confiance est mesurée grâce à une évaluation du comportement de l'acteur sur la base de son historique de collaboration [51, 189, 168, 56]. Nous considérons que la deuxième catégorie (*i.e.* la réputation) est plus adéquate pour notre contexte RSE, car la réputation va donner un aspect dynamique à notre mécanisme de contrôle d'accès.

Généralement, *la réputation est un jugement porté par autrui sur la qualité d'une personne* [103]. De nombreux travaux qui rentrent dans le cadre de la catégorie de la réputation sont basés sur des approches probabilistes. En effet, ces travaux se basent sur le *Beta model* [107, 149, 46, 183, 47]. Le modèle *Beta* utilise un paramètre (Θ) pour prédire le comportement, bon ou mauvais ($1-\Theta$), d'un acteur dans le futur, et ce par rapport à un comportement attendu. Dans [174], les auteurs essayent d'approcher la valeur du paramètre inconnu (Θ) en se basant sur l'historique des interactions de l'acteur. Les auteurs dans [165] proposent un modèle de contrôle d'accès sous forme d'extension du modèle ABAC. Ce modèle met l'accent sur l'aspect de confiance et de confidentialité pour l'administration des autorisations dans les systèmes collaboratifs de gestion de crise. En effet, la confiance est mesurée sur la base de trois dimensions, à savoir, le degré de collaboration entre l'organisation et le demandeur d'accès, les recommandations et la réputation. Cette dernière repose sur l'analyse

40. Par exemple numéro de carte bancaire, certification de compétence, une preuve d'appartenance à une organisation donnée, *etc.*

de l'historique d'accès de l'acteur pour augmenter, ou diminuer, sa réputation à travers le temps en fonction de paramètres subjectifs définis par l'entreprise. Cela signifie que la réputation d'un même acteur n'évolue pas de la même manière d'une entreprise à une autre.

Selon notre point de vue, il est plus intéressant d'exploiter l'évaluation de la confiance numérique reflétée par la réputation basée sur l'historique comportemental d'une manière plus ferme dans le contrôle d'accès. Plus précisément, la confiance doit être une contrainte essentielle dans les politiques de contrôle d'accès. Cependant la confiance ne doit pas être statique et doit pouvoir évoluer d'une manière dynamique à travers le temps, et ce sur la base de l'ensemble des actions passées (*i.e.* l'historique) réalisées par le sujet en question. Dans cette optique, il est possible que d'autres paramètres entrent en considération pour la construction de politiques basées sur la confiance numérique. Par conséquent, nous nous sommes tournés vers l'étude d'une gestion plus généralisée de la confiance avec d'autres contraintes contextuelles. Notre étude s'est ainsi focalisée sur les mécanismes de *gestion du risque*, car ces derniers représentent (par définition) les menaces de requêtes de demande d'accès et leurs impacts sur les enjeux présents dans le contexte collaboratif.

3.2.5.3 Gestion du risque

Aujourd'hui, la dépendance des entreprises aux technologies IT est devenue très importante [159]. Cette dépendance signifie que les entreprises doivent être en mesure d'identifier et de faire face aux nouvelles vulnérabilités et menaces auxquelles leurs systèmes sont exposés, et ce dans le but d'adapter leurs systèmes aux évolutions des environnements et des besoins organisationnels qui s'en suivent.

La gestion du risque pour une organisation est un processus complet composé de quatre facettes :

- ***frame risk*** : définir le contexte (*i.e.* environnement) du risque afin d'adopter une stratégie d'évaluation, de réponse et de supervision du risque.
- ***assess risk*** : l'évaluation du risque à travers, l'identification des menaces vis-à-vis des entreprises, les vulnérabilités du contexte défini, et les éventuels susceptibles dommages, *i.e.* impact sur la confidentialité des ressource dû aux menaces et aux vulnérabilités.
- ***respond to risk*** : réaction de l'entreprise basée sur les résultats de l'évaluation du risque.
- ***monitor risk*** : supervision fondée sur le maintien de connaissances sur les informations de sécurité, les vulnérabilités et les menaces afin d'appuyer les décisions basées sur le risque.

Dans le cadre de cette thèse, le contexte dans lequel nous évaluons le risque (*i.e. frame risk*) est le réseau social d'entreprise. Par ailleurs, les réactions des entreprises (*i.e. respond to risk*) vis-à-vis des résultats de l'évaluation du risque vont porter sur les décisions du contrôle d'accès en suspendant l'accès aux acteurs concernés. Quant au monitoring du risque, nous omettons cette tâche, car elle se base en général sur les résultats d'expérimentations réelles dont nous disposons pas à ce stade. Enfin, nous nous focalisons dans nos travaux sur le risque sur l'évaluation du risque (*i.e. risk assessement*) en identifiant les paramètres pertinents relatifs à notre contexte RSE pour l'évaluation du risque.

Le processus d'évaluation du risque requiert l'étude des aspects suivants :

- outils, techniques et méthodes d'évaluation,

- les hypothèses relatives à l'évaluation du risque,
- les contraintes susceptibles d'influencer l'évaluation du risque,
- la manière dont les informations sur le risque sont collectées, traitées et communiquées à travers les entreprises,
- la manière dont sont mesurées les menaces (sources et méthodes).

En général, le risque est défini par la probabilité qu'un événement se produise avec ses conséquences sur le système [151]. Dans le contexte de la sécurité informatique dans les entreprises, un événement est communément considéré comme une menace qui se base sur une ou plusieurs vulnérabilités de l'environnement informatique afin d'occasionner un impact négatif, *e.g.* la destruction, l'altération et le vol des informations de l'entreprise [140]. Par exemple, un attaquant vole des informations sensibles (*menace*) à travers une interface compromise (*vulnérabilité*), ce qui peut infliger des pertes à l'entreprise (*impact*) [87]. Dans ce sens, l'évaluation du risque se résume a la formule suivante ([6], [151]) : $risque = vulnérabilité \times menace \times impact \Longleftrightarrow f(V,M,I)$.

Le risque et la confiance sont les deux aspects sur lesquels nous nous sommes basés dans notre étude qui vise à donner un aspect dynamique au mécanisme de contrôle d'accès destiné au RSE. Dans ce qui suit, nous faisons le point sur les travaux de recherche établis dans cadre du risque et de la confiance numérique.

Revue de la littérature sur le risque et le contrôle d'accès

Le travail réalisé dans [112] propose un framework de contrôle d'accès basé sur le modèle ABAC et l'évaluation du risque. Le modèle est principalement fondé sur des besoins opérationnels, des facteurs de situations[41], des politiques adaptables de contrôle d'accès ainsi que des heuristiques basées sur les vulnérabilités identifiées dans les cas pratiques antérieurs. Ces aspects sont formalisés sur la base d'extensions UCON (usage control) [121, 156]. Cependant, cette proposition ne fournit pas des détails sur les méthodes d'évaluation ni d'implantation de chacun de ces paramètres. Par ailleurs, la définition du risque n'est basée sur aucun standard.

Dans [154], les auteurs proposent un système de contrôle d'accès qui repose sur une approche d'évaluation de risque. Le système calcule la valeur du risque (entre 0 et 1) pour chaque requête entrante d'un sujet sur un objet. Ce système ne permet pas d'empêcher une tentative d'accès qui reflète une valeur élevée de risque, il permet plutôt d'attribuer un accès temporaire avec des post-obligations que le sujet doit accomplir après l'autorisation accès. En effet, l'estimation du risque est basée sur un système à inférence floue, avec le niveau d'intégrité du sujet et de sensitivité d'objet. Cela signifie que la vulnérabilité du contexte n'est pas prise en considération.

Dans *Dimmock* [47], on considère que la confiance est étroitement liée au risque. Les auteurs soulignent l'importance de la considération de ces deux aspects dans les prises de décisions d'autorisation. Ils proposent une solution basée sur une fonction de densité de probabilité et la confiance pour estimer le niveau du risque de chaque interaction. Ainsi, l'accès est autorisé lorsque le niveau du risque est assez bas, ou la confiance est assez élevée. La confiance peut dans ce cas être consi-

41. i.e. conditions environnementales externes sous lesquelles sont basées les décisions de contrôle d'accès

dérée comme la menace. Cependant, pour s'adapter aux propriétés du RSE tout en respectant le standard de définition du risque, il est également important de considérer la *vulnérabilité* et l'*impact*. Par ailleurs, dans [67] est proposée une approche d'évaluation du risque basée sur la théorie de la décision dans l'incertain. Les auteurs combinent les probabilités de confiance avec les résultats pour mettre en place un système de détection de *Spam*.

Dans cette même optique qui lie la confiance au risque, les auteurs dans [178] proposent un système de contrôle d'accès basé sur une évaluation dynamique du risque. Le système proposé est une amélioration (extention) du modèle de contrôle d'accès multi-niveaux (MAC Bell-LaPadula [25]), où le niveau d'intégrité (confiance) de l'acteur et la sensibilité (impact) de la ressource jouent un rôle central. Ainsi, les auteurs proposent une méthode d'évaluation de la confiance et du risque, en considérant que la confiance est l'effet opposé du risque. En effet, basées sur l'historique d'interaction local du sujet et des recommandations externes, les méthodes d'évaluation de la confiance et du risque évaluent, respectivement, le pourcentage des bonnes et mauvaises interactions par rapport au reste, en les pondérant avec l'évolution de la confiance du sujet à travers le temps. Les décisions sont par la suite prises sur la base de la comparaison directe de la confiance et du risque : si la confiance est supérieure au risque, le sujet sera autorisé, sinon, sa requête sera rejetée. Les auteurs proposent, dans le même travail, une deuxième méthode qui se focalise plus l'historique d'interactions du sujet en donnant plus d'importance aux interactions les plus récentes par rapport aux anciennes, et ce grâce à la méthode *Exponentially Weighted Moving Average (EWMA)* [59]. Le principal souci avec ce travail dans le cadre d'un RSE réside dans la formule d'évaluation de la confiance et du risque. Cela est dû à la nature sensible des ressources et l'ouverture laxiste de l'environnement social. En effet, pour une telle configuration d'environnement collaboratif, nous avons besoin d'intercepter rapidement tout changement brusque et suspicieux de comportement et de réagir très rapidement pour réduire la menace. Ces objectifs ne sont pas couverts par la formule proposée dans [178].

L'article [70] traite le contrôle d'accès pour le partage de ressources dans les fédérations de Clouds [118] en essayant de compenser les cas d'absence de fédération d'identité [122], et ce au moyen de l'évaluation du risque. En effet, le mécanisme présenté se base sur le modèle ABAC et une architecture XACML. Le mécanisme d'évaluation du risque (intégré au niveau PDP) s'active dans le cas où le système ne trouve pas de fédération d'identité entre le domaine (cloud) du sujet demandeur d'accès le domaine étranger où est hébergée la ressource demandée. Ce système vise à promouvoir le passage à l'échelle et traiter des requêtes particulières dépendant fortement du contexte.

Cependant, l'évaluation du risque se fait au niveau de chaque domaine d'une manière indépendante et éventuellement différente des autres domaines. Cela ne s'adapte pas vraiment au contexte du RSE, car afin de ne pas brider la collaboration, nous avons besoin d'un mécanisme généralisé d'évaluation du risque qui respecte l'autonomie de chaque domaine (entreprise) en matière d'administration (*i.e.* le paramétrage) du risque.

L'efficacité d'un mécanisme de contrôle d'accès est relative au contexte de son application. Ainsi, bien que nous soutenons l'importance du contrôle d'accès dynamique, nous n'adhérons pas à l'idée de remplacer un mécanisme de contrôle d'accès ferme et compact par un mécanisme de risque basé sur des métriques de probabilité, *i.e.* estimation approximative de la menace et son impact sur le système.

De plus, les décisions du risque et de confiance sont censées améliorer et soutenir les décisions d'autorisation, mais elles ne doivent en aucun cas atténuer l'effet des politiques de contrôle d'accès. Par conséquent, le mécanisme d'évaluation du risque et de confiance doit être léger, dans le sens où son intégration à un mécanisme de contrôle d'accès doit être facilement paramétrable, voire même désactivable. Par conséquent, il doit se situer à un niveau parallèle et indépendant de celui des politiques initiales de contrôle d'accès, et comme un mécanisme complémentaire de décision.

Nous avons étudié jusqu'à présent les principaux travaux concernant les modèles classiques (*i.e. MAC*, *DAC* et *X-RBAC*), le modèle *ABAC* ainsi que le *contrôle d'accès dynamique* qui se base sur le contexte en incluant la confiance numérique et la gestion du risque. Dans ce qui suit, nous allons étudier un aspect particulier du contrôle d'accès lié aux environnements collaboratifs en général et aux RSEs en particulier, à savoir la délégation. La délégation consiste principalement en un transfert temporaire de droits d'accès à des ressources.

3.2.6 Délégation

La délégation est un mécanisme de transfert de droits, connexe au contrôle d'accès. Généralement, une délégation peut se produire selon deux formes [58], une délégation via un administrateur et une délégation via un utilisateur. La différence est relative aux droits possédés vis-à-vis des droits délégués. Un utilisateur est obligé d'avoir le privilège pour pouvoir le déléguer, contrainte qui ne concerne pas l'administrateur. Nous nous focalisons dans cette thèse sur la délégation via l'utilisateur. Par ailleurs, une délégation peut être entre humains ou entre machines (*e.g.* agent logiciel, service web, *etc.*) ou les deux [24].

Il existe plusieurs définitions dans la littérature du concept de délégation. *Zhang et al.* [205] voient la délégation comme suit : *"l'objectif de la délégation est d'avoir le travail fait en l'envoyant à quelqu'un d'autre, par exemple, un subordonné"*. *McNamara et al.*[141] considèrent la délégation comme *"la marque d'une bonne supervision"*. Dans le cadre du contrôle d'accès, le besoin de délégation se pose quand un utilisateur a besoin d'agir pour le compte d'un autre utilisateur pour accéder à ses ressources autorisées. Cela pourrait être pour un temps limité, par exemple des vacances, ou bien pour un remplacement, *i.e.* toujours agir en cas d'absence de l'acteur principal. Ainsi, une délégation permet à un *délégataire* d'acquérir des privilèges lui permettant de réagir dans des situations ou d'accéder à des informations sans besoin de se référer à son *délégant* [205]. Cependant, dans une délégation il existe quelques spécificités, notamment dans les mécanismes à base de rôles, à savoir [75] :

- une délégation peut être *totale* ou *partielle*. Dans une délégation totale, un utilisateur délègue toutes ses permissions, relatives à son rôle par exemple. Tandis qu'avec une délégation partielle, l'utilisateur peut déléguer une partie de ses privilèges.
- délégation transitive multi-niveaux. Cette propriété permet un utilisateur de redéléguer les droits qui lui sont délégués tout en respectant une certaine profondeur (prédéfinie) de redélégation.
- une délégation peut être du type *"grant"* ou *"transfert"*. La différence est que dans une dé-

légation *grant*, les droits d'accès seront partagés entre les deux parties de délégation (*i.e.* délégataire, délégant), dans le sens où le délégant continue de pouvoir exploiter ses droits d'accès délégués. En revanche, dans le cas d'une délégation *transfert*, le délégant n'a plus le droit d'accomplir les tâches relatives aux droits délégués.

- dans le cas d'une délégation de droits reliés à un rôle faisant partie d'une hiérarchie, le délégataire se voit procurer tous les privilèges des rôles subordonnés du rôle délégué.

Par ailleurs, une délégation efficace requiert certaines exigences, à savoir la révocation et les conditions. La révocation est un aspect très important qui doit accompagner toute délégation. En effet, une délégation à un moment donné doit pouvoir être suspendue, *i.e.* validité temporaire des droits. Une procédure de révocation consiste à enlever les privilèges délégués, ou de revenir à l'état d'avant la délégation. Quant aux conditions, elles concernent les délégations et les révocations. Une condition spécifie des restrictions sous forme de contraintes. Ces contraintes doivent être satisfaites au moment de la validation de la délégation respectivement la révocation. Dans ce qui suit, nous allons présenter quelques travaux portant sur la délégation dans les environnements collaboratifs.

Délégation et travaux connexes

Beaucoup de travaux se sont intéressés au concept de la délégation. La plupart des travaux qui ont contribué à l'émergence du concept, tels que [86, 11, 82] , se sont plus au moins focalisés sur la délégation homme-machine et machine-machine [24]. Vu que dans la pratique, RBAC a été le formalisme dominant dans le domaine du contrôle d'accès pendant une bonne époque [101], beaucoup de travaux concernant la délégation se sont tournés vers le formalisme à base de rôle.

Les modèles à base de rôles les plus populaires dans le domaine de délégation sont *RBDM0* (Role-Based Delegation Model 0), et ses variantes, *RBDM1* et *RDM2000* [24, 23, 22, 205]. Ces modèles couvrent plusieurs aspects relatifs à la délégation, spécialement les hiérarchies de rôles ainsi que les délégations multi-niveaux. Cependant, la délégation partielle n'est pas supportée dans ces modèles, dans le sens où le rôle (et les permissions qui en découlent) est la plus petite unité dans une délégation. Dans notre contexte social et hétérogène, les délégations partielles sont très importantes car le même rôle dans deux domaines (entreprises) différents peut ne pas donner droits aux mêmes privilèges.

Par ailleurs de nombreuses extensions RBAC orientées vers la gestion du contexte traitent l'aspect de délégation. [105] est une extension du modèle GTRBAC qui supporte les délégations dans des hiérarchies hybrides de rôles ainsi que différents types de délégation comme les délégations partielles. Une hiérarchie hybride permet de combiner l'aspect héritage et activation de rôle dans une hiérarchie de rôles. [161] permet de prendre en considération plus de types de délégation, et ce, autour des rôles et des permissions avec des contraintes temporelles et/ou géographique. [128] traite également la délégation à l'égard des hiérarchies de rôle.

Un des modèles de délégation à base de rôles les plus récents est connu sous le nom *GemRBAC* [75]. *GemRBAC* se présente comme le framework qui englobe les extensions les plus importantes du modèle RBAC. Fondé sur une représentation UML, GemRBAC est implanté en *Object Constraint*

Language (OCL). Plus précisément, les composants des politiques RBAC comme le rôle, la délégation, les permissions, les sessions, sont représentés par des classes constituées d'attributs nécessaires permettant de couvrir tous les aspects de contrôle d'accès à base de rôle, comme la séparation des devoirs, les hiérarchies des rôles, les pré/post conditions. En fait, grâce aux attributs constituant les classes du modèle, ce modèle peut être vu comme une version ABAC de RBAC avec une implantation qui tire profit des avantages d'expressivité du standard *OCL*. Sur le plan technique, les auteurs de GemRBAC le présentent comme étant un modèle générique capable de s'adapter à n'importe quelle entreprise et facilement configurable par les administrateurs en se basant sur les Checkers OCL disponibles. Cela signifie que le langage nécessite une certaine expérience dans le domaine du contrôle d'accès pour l'administration des politiques d'accès aux ressources partagées, contrainte assez difficile à satisfaire dans le cadre des réseaux sociaux et des approches user-centric de définition de règles.

XACML permet également de modéliser des règles de délégation grâce à la flexibilité et la richesse des attributs. Dans [191], les auteurs ont développé la langage de politique *BelLog* pour exprimer les délégations ainsi que la composition de politiques avec la capacité de raisonnement. Dans [176] est proposé un modèle de politique d'administration et de délégation fondé sur XACML et le serveur *Delegent* [78]. Ce modèle est capable d'exprimer les chaînes de délégation, avec également des contraintes sur les délégations en utilisant le langage XACML.

XACML 3.0 introduit le concept de délégation dynamique. En effet, une délégation dynamique permet à certains utilisateurs de créer des politiques de durée limitée afin de déléguer certaines capacités à d'autres utilisateurs [155, 73]. Contrairement à *XACML 2.0*, la notion de circonstance dans l'administration des politiques (*i.e.* le contrôle de la création et la modification des politiques) n'était pas abordée. Par exemple, dans *XACML 3.0* un utilisateur peut créer une règle permettant à un autre utilisateur de faire des tâches pour son compte tant qu'il sera en vacances, alors que dans XACML 2.0 un utilisateur pouvait juste attribuer ses droits à un autre sujet. Dans [64], les auteurs proposent un framework pour l'amélioration du profil de délégation XACML grâce aux ontologies. Les opérations de délégation ainsi que de vérification et de révocation peuvent être effectuées selon les entités impliquées dans la délégation et le flux généré par l'instanciation des classes de l'ontologie et de leurs interrelations [64]. Le système est complètement automatisé, car les opérations de délégation sont réalisées grâce à un algorithme qui ne nécessite aucune intervention humaine.

Dans le même cadre des ontologies, les auteurs dans [109] présentent un framework de délégation pour systèmes multiagents. Ce modèle utilise une ontologie basée sur une base de connaissances et des politiques écrites en *Prolog*. La délégation entre agents dans l'environnement multi-organisationnel se fait en utilisant de la confiance mutuelle entre agents afin de former les chaînes de délégation. Cependant, l'évaluation de la confiance n'est pas détaillée et, dans ce travail, elle n'a pas été intégrée dans l'évaluation de la règle de la délégation. Bien que des étiquettes pour désigner si un objet peut être redélégué soient utilisées, ainsi que des agents de sécurité pour contrôler les agents qui s'entre délèguent les ressources, le problème avec les chaînes reste un peu similaire au contrôle d'accès discrétionnaire, dans le sens où cela peut amener le système dans un état d'insécurité à cause de fuites d'informations, par exemple, dans le cas où un des agents de sécurité est

compromis.

3.2.7 Vers une implantation formelle de XACML

Comme nous venons de le souligner dans la section **cf.** *3.2.4.2*, de nombreux travaux ont exploré l'idée de donner un formalisme logique au langage XACML. Pour la mise en oeuvre de notre modèle de contrôle d'accès dynamique, nous avons fait le choix d'utiliser la logique temporelle de premier ordre *Event-Calculus* [116, 148], que nous allons maintenant présenter.

Event-calculus

Event-calculus est un langage formel pour représenter et raisonner sur des systèmes dynamiques [21]. Le formalisme logique de premier ordre Event-calculus inclut une arithmétique pour la gestion du temps d'occurrence d'événements. L'occurrence d'un événement (également appelé action "$\mathscr{A}$") provoque un changement de l'état d'un (ou plusieurs) attribut(s) de l'environnement. Le terme "Fluent" "$\mathscr{F}$" est utilisé pour représenter les états du système. Il s'agit d'une propriété binaire du contexte qui, à travers le temps "$\mathscr{T}$", prend les valeurs, *"vrai"* ou *"faux"*.

Event-calculus utilise des prédicats, écrits en clauses de *Horn* [52], pour la gestion des événements et leurs fluents. Les principaux prédicats d'*Event-calculus* sont :

- $\boldsymbol{Initiates(e,f,t)}$: à partir du l'instant $t+1$ l'état du fluent f prend la valeur *vraie* si l'événement e se produit à l'instant t.
- $\boldsymbol{Terminates(e,f,t)}$: après l'occurrence à l'instant t de l'événement e, l'état du fluent f est d'ores et déjà changé à *faux*.
- $\boldsymbol{Happens(e,t)}$: forcer et/ou indiquer l'occurrence de l'événement e à l'instant t.
- $\boldsymbol{HoldsAt(f,t)}$: vérifier et/ou indiquer l'état (*vrai*/*faux*) du fluent f à l'instant t.
- $\boldsymbol{InitiallyTrue(f)}$: l'état du fluent f est initialement (l'instant 0) *faux*.
- $\boldsymbol{InitiallyFalse(f)}$: l'état du fluent f est initialement (l'instant 0) *vrai*.
- $\boldsymbol{Clipped(t_1,f,t_2)}$: l'état du fluent f change de *vrai* à *faux* durant l'intervalle du temps $[t_1,t_2]$.
- $\boldsymbol{Declipped(t_1,f,t_2)}$: l'état du fluent f change de *faux* à *vrai* durant l'intervalle du temps $[t_1,t_2]$.
- $\boldsymbol{Trajectory(f_1,t_1,f_2,t_2)}$ [179] : si le fluent f_1 est initialisé à l'instant t_1, alors le fluent f_2 devient *vrai* à l'instant t_1+t_2.

Event-Calculus supporte plusieurs modes de raisonnement, à savoir, déductif, inductif et abductif [166]. Le raisonnement déductif utilise la description du comportement du système avec l'historique des événements qui se produisent dans le système afin de dériver les fluents qui sont initialisés (vrai) à un instant donné. L'induction permet de dériver la description du comportement du système à partir d'un événement historique donné et des informations sur les états de fluents qui sont validés (vrai) à des instants différents [21]. Dans notre mécanisme de contrôle d'accès, nous nous sommes basés sur le mode de raisonnement abductif[42]. Par définition, le raisonnement abductif cherche à trouver les causes des situations. L'utilisation du raisonnement abductif dans *Event-calculus* consiste d'abord à spécifier en amont l'état initial de l'environnement du système avec l'éventuel but(s) at-

42. Discuté par *Charniak and McDermott (1985,chap.8), Shanahan (1989 ;1997b, chap.17 ; 2000a), Denecker, Missiaen, et Bruynooghe (1992) et Hobbs, Stickel, Appelt et Martin (1993)* [147].

tendu(s). Ensuite, suivant le raisonnement abductif, on vérifie si le but est satisfait à un instant précis, et ce à travers un schéma basé sur l'occurrence d'une suite d'événements. En d'autres termes, étant donné un état initial de l'environnement, le raisonneur détermine la séquence d'événements dont l'occurrence est nécessaire pour atteindre, à un instant précis, un objectif initialement prédéfini. Cela correspond parfaitement à notre vision de politiques de sécurité et leurs schémas d'autorisation. En effet, nous considérons les politiques comme étant une prescription dans laquelle un objectif de sécurité est vérifié à travers le comportement du système. Ce dernier est, à tout instant, dans un état donné qui est susceptible de changer suite à l'occurrence de certains événements, éventuellement contrôlés.

Pourquoi Event-Calculus ?

Nous croyons vigoureusement qu'un formalisme logique est plus approprié pour l'implantation des politiques de sécurité, notamment dans le contexte des réseaux sociaux où les politiques sont définies par des utilisateurs majoritairement humains. En effet, le formalisme logique offre au système une ouverture au raisonnement et à l'analyse de satisfaction des formules, ce qui permet de vérifier la consistance des politiques définies. Cette vérification peut bien évidemment être automatique lorsque le formalisme est doté d'un outil de raisonnement, à l'image des solveurs *SAT*. *Event-Calculus* dispose d'un tel outil de raisonnement, appelé *Dec-reasoner*. *Dec-reasoner* est conçu pour raisonner sur la satisfaction des prédicats *Event-Calculus*. Ainsi, les politiques deviennent plus précises et expressives, avec une représentation flexible et non ambiguë. En outre, la vérification des conflits entre politiques ainsi que leurs validations (en vue de la prise de décision) devient facile et pratique.

Event-Calculus a été initialement utilisé par *Bandara et al.* dans [21] pour la modélisation du contrôle d'accès. Pour nous, la principale motivation derrière l'utilisation de ce formalisme de logique de premier ordre est la prise en charge de l'aspect temporel, ce qui nous a permis de rendre notre modèle de décision dynamique. Plus précisément, les décisions des politiques sont évaluées de manière continue car l'effet (permit/deny) de chaque instance de politique constitue un état d'un l'objet (la politique) dans l'environnement, qui peut changer à tout moment suite à l'occurrence de certains événements. Par conséquent, l'accès d'un acteur autorisé par la politique initiale peut être suspendu à tout moment. En outre, l'aspect temporel nous a permis de gérer les permissions temporaires, dites aussi délégation, et ce d'une manière auto-révocable. En effet, *Event-Calculus* inclut une arithmétique de gestion du temps, ainsi que des prédicats (à l'image de *Trajectory*) qui permet de paramétrer le changement d'un état donné de l'environnement vers un nouvel état, et ce après un intervalle de temps prédéfini. Grâce à cela, les permissions temporaires ne constituent plus un problème pour la formalisation logique.

DEC-reasoner

Il existe trois outils de raisonnement pour Event-Calculus [43], à savoir, *Event calculus answer set programming*, *Discrete Event Calculus Reasoner*, et *Discrete event calculus theorem proving*. Nous nous

43. http ://decreasoner.sourceforge.net/csr/index.html

intéressons au *Discrete Event Calculus Reasoner (DEC-reasoner)* [44] [146] qui est fondé sur des solveurs SAT (Relsat, Walksat et MiniSat) et inclut le mode de raisonnement abductif. Ainsi, nous nous basons sur le *DEC-Reasoner* et le solveur Relsat pour la vérification et la validation de politiques de sécurité de notre modèle de contrôle d'accès. DEC-Reasoner supporte aussi :

- *the commonsense law of inertia* : le principe de loi de l'inertie dans lequel un état du système persiste à travers le temps tant qu'il y a pas d'événement qui permet de l'arrêter et de mettre le système dans un autre état appartenant à un ensemble limité de situations, *i.e.*, le problème du cadre.
- des contraintes sur les changements d'état (causés par l'occurrence d'événement) de l'environnement,
- *release from the commonsense law of inertia* : libération du système d'un état d'inertie initial.
- effets et événements indirects : un état peut conduire à d'autre(s) état(s), ou le déclenchement d'un nouvel événement peut être provoqué par d'autre(s) événement(s) et/ou état(s),
- événements non déterministes : étant données les différentes situations susceptibles de découler d'un événement, il y a cependant aucune garantie du résultat final obtenu après l'occurrence de l'événement en question,
- aspect concurrentiel des événements : plusieurs événements peuvent se produire au même moment provoquant l'accumulation ou l'annulation de certains effets dans l'environnement.

Quand le raisonneur *DEC-Reasoner* est invoqué, il transforme la description du domaine en un problème de satisfaisabilité booléenne (SAT). Grâce aux solveurs SAT, *DEC-Reasoner* donne une ou plusieurs solutions sous forme de modèle(s). Un modèle est décodé en fonction de la description du domaine et affiché pour indiquer l'état du système à chaque instant appartenant à l'intervalle temporel de raisonnement, et ainsi indiquer si le but a été satisfait (abduction). Par ailleurs, *DEC-Reasoner* est un outil open source ouvert aux contributions externes en vue de son amélioration. Un exemple est la fonction *DictHash* pour l'encodage SAT qui a été améliorée par *Ehtesham Zahoor* dans [200, 202]. La fonction *DictHash* améliorée réduit considérablement le temps de l'encodage SAT.

3.3 Conclusion

Dans ce chapitre, nous avons introduit les concepts fondamentaux de la gestion des identités numériques et leurs accréditations. Nous avons présenté ces différents concepts de gestion d'authentification, et de contrôle d'accès (autorisation et monitoring) avec un état de l'art portant sur de nombreux travaux (des standards, des protocoles et des solutions) qui ont été élaborés et appliqués dans des contextes proches du nôtre. Nous avons également discuté les avantages ainsi que les limites que présentent ces travaux vis-à-vis de notre contexte RSE. Nous avons par ailleurs introduit certains concepts complémentaires que nous avons utilisés dans la conception et la mise en œuvre de nos différents composants (AAA) de sécurité pour *OpenPaaS* RSE. Dans le chapitre suivant, nous

44. http ://decreasoner.sourceforge.net/

détaillerons nos contributions sur la gestion de l'authentification, de l'autorisation et de l'audit des échanges collaboratifs au sein du RSE *OpenPaaS*.

3.4 Synthèse de l'état de l'art

Dans ce chapitre nous avons présenté un état de l'art sur les différentes parties de la problématique que nous avons traitée (**cf. chapitre Problématique et motivations**), à savoir la gestion des identités numérique et leurs autorisations dans les environnements collaboratifs et distribués de type réseau social d'entreprise (RSE).

Dans le cadre de notre étude des travaux sur l'authentification, nous nous sommes focalisés sur la question du mode de gestion des identités numérique collaboratives dans un contexte RSE. À ce propos, nous avons cité de nombreux outils d'authentification comme l'authentification forte (Fido [77]), l'authentification unique SSO (OpenID [162], OAuth [110]) et l'authentification fédérée (SAML [90], WS-federation [57]). Après l'étude de ces nombreux travaux, nous avons souligné les avantages que procurent les mécanismes d'authentification SSO. Cependant, la question qui reste ouverte par rapport à notre contexte hétérogène RSE est relative à l'interopérabilité en matière de mécanismes d'authentification utilisés par les entreprises collaboratives. En effet, un de nos objectifs est d'offrir à chaque entreprise la possibilité de préserver son mécanisme d'authentification dans le cadre collaboratif, *i.e.* une authentification interentreprises. La difficulté réside dans le fait que les mécanismes d'authentification utilisés par les entreprises sont incompatibles les uns avec les autres.

Concernant la partie autorisation, nous nous sommes penchés sur les principaux travaux portant sur le contrôle d'accès dans les environnements collaboratifs, comme MAC/DAC, RBAC/ Or-BAC et ABAC. Nous avons étudié les avantages ainsi que les limites de ces modèles par rapport à notre contexte RSE et nos besoins de sécurité qui se résument à :

- la flexibilité du modèle de règle d'autorisation,
- l'interopérabilité en matière de règle d'autorisation,
- la fluidité d'échange de ressources collaboratives,
- l'autonomie de l'entreprise,
- la dynamicité des règles d'autorisations.
- la gestion des permissions temporaires, *i.e.* délégations.

La flexibilité du modèle d'autorisations réside dans la possibilité de rajouter des contraintes à une règle de contrôle d'accès définie sur la base de ce modèle. Après l'étude des solutions existantes, nous avons constaté que le modèle ABAC est le plus adéquat pour répondre à un tel besoin grâce à la liberté de définition des attributs. Les attributs permettent par ailleurs d'homogénéiser les sémantiques de définition des profils des sujets des règles [45] de contrôle d'accès entre plusieurs entreprises, *i.e.* l'interopérabilité.

La fluidité d'échange de ressources est l'un des besoins essentiels d'un environnement ubiquitaire tel qu'un RSE. En effet, nous avons constaté que l'approche *user-centric* [136, 88, 92] pour le partage

45. composées initialement de sujet, objet et action.

de ressources est très intéressante pour la fluidité des échanges collaboratifs. À l'image du modèle DAC, dans un partage *user-centric* le partage de ressources se fait par l'utilisateur. Cela signifie que même l'autorisation d'accès à la ressource partagée va être définie par l'utilisateur. Cependant, les autorisations de partage de ressources peuvent être en contradiction avec les politiques de l'entreprise propriétaire des ressources à partager. Pour répondre à ce besoin, nous avons souligné la nécessité de pouvoir combiner deux niveaux de politiques, à savoir les politiques de partages définies au niveau de l'utilisateur et les politiques d'entreprise définies au niveau de l'entreprise. Pour ce faire, nous avons réalisé que le langage XACML est le plus adapté à une telle modélisation de politique. De plus, le langage XACML est le plus utilisé pour l'implantation du modèle ABAC, *e.g.* les projets *Cardea* [123], epSOS [7], NHIN [8].

Dans un environnement riche en interactions tel qu'un RSE, les politiques entreprise ne peuvent pas être de la même fréquence ni du même niveau d'abstraction que celles des utilisateurs. D'une part, une entreprise ne peut pas définir une nouvelle règle pour chaque nouveau partage de ressource, et d'autre part, une règle d'entreprise ne peut pas prendre la forme d'une règle de partage (*i.e.* sujet, objet et action). Les politiques d'entreprises doivent ainsi être définies avec un niveau d'abstraction plus élevé tout en étant capable de contrôler chaque règle de partage. Pour répondre à ces besoins, nous nous sommes tournés vers le contrôle d'accès dynamique [178] pour la définition des politiques d'entreprise. Nous avons étudié la gestion du risque [151] ainsi que la confiance numérique [12] et la réputation [107] dans le cadre de la supervision des comportements des acteurs de collaboration.

Par ailleurs, le partage *user-centric* exige une vigilance dans la définition des politiques de partage de ressources. En effet, un utilisateur peut définir une règle d'une politique de contrôle d'accès non correcte, ce qui va provoquer des incohérences au moment de la combinaison avec une politique d'entreprise. Pour répondre à ce besoin de vérification de la cohérence des politiques, nous avons étudié plusieurs solutions basées sur XACML [164]. Cependant nous avons constaté que la vérification basée sur XACML est coûteuse en temps et en traitement notamment lorsqu'il s'agit d'un environnement ubiquitaire tel qu'un RSE. Par conséquent, nous nous sommes penchés sur les formalismes logiques [114, 45, 138] pour la vérification automatique de la cohérence des politiques de contrôle d'accès.

Dans notre étude des formalismes logiques pour la définition d'un modèle de politique de contrôle d'accès, nous avons pris en considération deux paramètres essentiels par rapport à nos objectifs de sécurité au sein du RSE, à savoir le calcul d'événements [91] ainsi que le temps. Le calcul d'événements nous permet d'avoir un contrôle d'accès dynamique basé sur l'occurrence en temps réel d'événements dans l'environnement collaboratif. La gestion du temps nous permet de gérer les autorisations temporaires auto-révocables (les délégations) [205]. Dans les travaux offrant un formalisme logique au langage XACML que nous avons étudiés, ces deux paramètres (les événements et le temps) ne sont pas pris en considération. Par conséquent, nous avons choisi un autre formalisme logique qui prend en considération le calcul d'événements et la gestion du temps pour la modélisation de notre propre modèle de sécurité ABAC avec une implantation XACMl. Il s'agit de la logique du premier ordre *Event-Calculus* [147], avec comme raisonneur l'outil *Dec-reasoner* [46] pour la vérification automatique

46. *http://decreasoner.sourceforge.net/*

de la cohérence des politiques ainsi que la prise des décisions d'accès.

Chapitre 4

Architecture et gestion des identités numériques

Sommaire

4.1 Introduction

Le premier axe de recherche de cette thèse concerne la conception d'une architecture de sécurisation adaptée aux communautés RSE. En effet, l'idée d'une collaboration professionnelle à travers une communauté RSE consiste à regrouper diverses disciplines et compétences dans un cadre collaboratif et coopératif commun. À titre de rappel, *une communauté est une entité logique faisant l'objet d'un cercle commun regroupant plusieurs entités collaboratrices passives et/ou actives*. Quant au RSE, c'est le cercle qui regroupe l'intégralité de ces communautés de collaboration. Par conséquent, nous avons besoin de définir l'architecture de collaboration dans ces environnements collaboratifs. Cette architecture va déterminer la manière dont seront gérées les entités collaboratives (actives et passives) au sein d'une communauté. Dans la section suivante, nous présentons les différents types de collaborations utilisés dans les environnements collaboratifs distribués.

4.2 Gestion et administration des identités et ressources

Notre vision du réseau social social d'entreprise est fondée sur le principe de collaboration professionnelle entre des *entreprises* et les personnes qu'elles regroupent. En effet, une entreprise peut être

vue comme un ensemble logique regroupant des ressources et des identités d'acteurs collaboratifs. Par conséquent, pour qu'un acteur puisse interagir (*i.e.* partager et/ou consommer des ressources) avec d'autres acteurs au sein d'une communauté RSE, il a besoin d'être représenté par une entité qui permettra de référencer ses informations identitaires ainsi que ses ressources destinées à la collaboration.

Les gestion et l'administration de la sécurité dans une communauté peut être définie selon plusieurs dimensions à savoir, la gestion des ressources, la gestion des politiques et la gestion des acteurs.

4.2.1 Gestion des ressources

Afin de permettre à chaque entreprise de préserver son autonomie sur la gestion des ressources collaboratives, et de promouvoir en outre le passage à l'échelle dans le processus de partage de ressources, nous avons choisi un mode de partage qui se base sur la virtualisation des ressources.

Plus précisément, les ressources physiques restent hébergées au niveau des serveurs de l'entreprise. Avant d'être partagée, chaque ressource est virtualisée dans la communauté RSE. Dans ce contexte, la virtualisation consiste à générer lien (logique) qui permet d'accéder à la ressource physique au niveau du serveur de l'entreprise depuis la communauté. Les informations concernant la virtualisation des ressources peuvent être stockées au niveau d'un module appelé *VGDS (Virtual Global Directory Service)* [17]. Le VGDS contient des identifiants de virtualisation faisant référence aux chemins des ressources physiques hébergées sur les serveurs de l'entreprise. Le VGDS sera ainsi à disposition de la communauté où seront déployées les ressources en question. En récapitulant, au sein d'une communauté donnée, une requête de demande d'accès établie par un acteur se fait vis-à-vis d'un identifiant de ressource virtualisée. Ensuite, une fois les droits d'accès vérifiés par le mécanisme de contrôle d'accès de la communauté, l'accès à la ressource physique aura lieu au niveau du serveur de l'entreprise propriétaire en question, comme cela est illustré dans la figure 4.1.

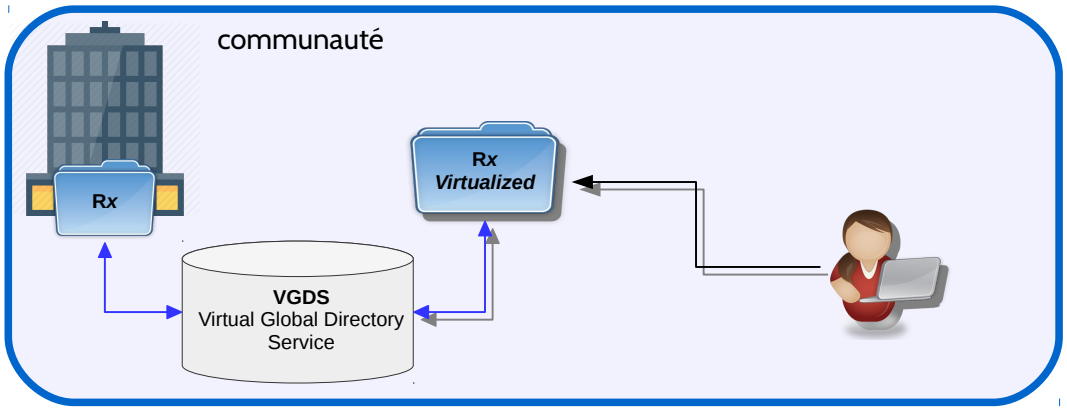

FIGURE 4.1 – Virtualisation des ressources collaborative(VGDS).

4.2.2 Gestion des politiques

D'un point de vue architecture, une politique est considérée comme une ressource particulière. Pour la gestion de politiques (figure. 4.2), nous avons mis en place une base commune de politiques au niveau de chaque communauté RSE, (*i.e.* centralisée). Cette base de politiques est administrée de manière autonome et indépendante par toute entité collaboratrice active, à savoir les acteurs et les entreprises. L'administration de la base de politiques consiste à ajouter, modifier et/ou supprimer des politiques (ensemble de règles) de contrôle d'accès aux ressources partagées. L'ensemble des politiques est mis en application (exécution) par un mécanisme de contrôle d'accès central au niveau de la communauté. Cette conception d'architecture a pour objectif de répondre au besoin d'interopérabilité grâce à un modèle homogène de politiques à base d'attributs identitaires d'une entité active. En outre, sachant que le contrôle d'accès se fait d'une manière centralisée au niveau de la communauté RSE (regroupant plusieurs entreprises) cette architecture permet de promouvoir la fluidité et l'agilité pour le partage des ressources.

Par ailleurs, afin de tirer avantage de la flexibilité de la collaboration ad-Hoc (*i.e.* de courte durée) nous proposons une approche de définition de règles centrée sur l'acteur collaboratif, une approche qui est cohérente avec la nature des réseaux sociaux en général. En effet, lorsqu'un acteur partage une ressource donnée avec un autre acteur au sein de la même communauté, il définit une règle de contrôle d'accès. Cette règle contient principalement quatre attributs, à savoir la ressource, le sujet qui va la consommer, l'action autorisée sur la ressource, et éventuellement la durée de mise à disposition. L'ensemble de ces attributs est ensuite stocké sous forme d'une règle au niveau de la base commune des politiques de contrôle d'accès de la communauté.

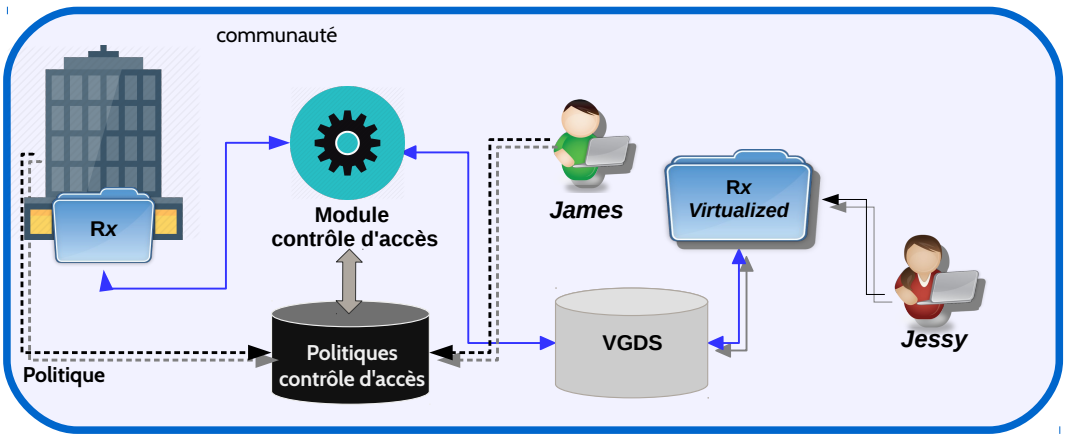

FIGURE 4.2 – Contrôle d'accès dans les communautés de collaboration.

Enfin, il est également important de respecter la propriété du faible couplage dans un environnement RSE, pour que l'entreprise garde son autonomie sur les ressources partagées par ses acteurs

(*i.e.* personnel). Par conséquent, nous avons conçu notre modèle de contrôle d'accès de telle sorte qu'il soit capable de combiner plusieurs politiques issues de plusieurs entités actives (y compris les entreprises). Ainsi, la base de politiques de la communauté peut aussi bien être gérée par les entreprises que par les acteurs collaboratifs. De plus amples détails à ce propos seront présentés dans le chapitre **cf.Contrôle d'accès**.

Pour résumer, comme le montre la figure 4.2 un acteur demandeur d'accès accède à une ressource (hébergée sur le serveur de l'entreprise propriétaire) à travers un lien de la ressource (virtualisée) généré par le VGDS. L'accès est soumis à une procédure de vérification de droits assurée par un module de contrôle d'accès qui se base sur les politiques définies par l'entreprise propriétaire et l'acteur interne (*James*) à cette entreprise qui partage la ressource en question avec le demandeur d'accès (*Jessy*).

4.2.3 Gestion des identités numériques

Vu que chaque communauté est indépendante, nous proposons que tout acteur possède une identité propre à chaque communauté. Comme le montre la figure 4.3, dans notre conception chaque acteur possède une identité initiale *"ID interne"* reliée à son entreprise. Cet identifiant interne sera le noyau des identifiants externes qui seront créés (en interne [47]) en vue de leurs exportations au niveau des communautés dont fera partie l'acteur en question. La liaison entre un identifiant externe et son identifiant interne au niveau de son fournisseur d'identité initiale est assurée par le biais de la ***fédération***.

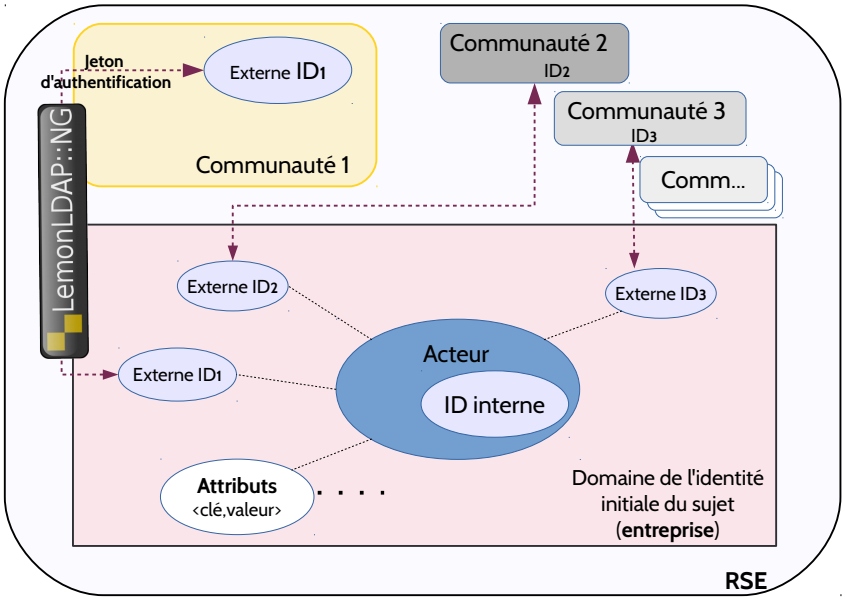

FIGURE 4.3 – Gestion des identité dans les communautés fédérées.

47. Au niveau de l'entreprise de l'acteur en question.

4.2.3.1 Fédération des identités numériques

Quand un acteur rejoint une communauté donnée, il y crée son identité relative à des attributs demandés par le gestionnaire d'identités de la communauté en question. Les valeurs des attributs seront dans un premier temps vérifiés auprès du fournisseur d'identités de l'acteur en question dans le cadre de la fédération. Après la validation des attributs, le fournisseur d'identités de l'entreprise génère un identifiant externe, le relie à l'identifiant interne de l'acteur, ensuite l'envoie à la communauté en question. Au niveau de la communauté ciblée, l'identifiant externe est relié en tant que principal *ID* (*i.e.* clé primaire) à l'ensemble des attributs fournis par l'acteur et validés par son fournisseur d'identités (entreprise). Une fois le processus d'inscription finalisé, l'acteur sera confronté à une procédure d'authentification à chaque fois qu'il souhaite rejoindre la communauté en question.

Le processus d'authentification vérifie l'authenticité des attributs qui forment l'identité d'un acteur. Dans un contexte distribué, le processus d'authentification implique en général trois entités, à savoir : l'acteur, le fournisseur d'identités et le fournisseur de services [32]. L'acteur est le sujet qui cherche à prouver la légitimité de son identité afin d'accéder à une ressource protégée. Le fournisseur d'identités est l'entité qui stocke et gère l'identité de l'acteur. Le fournisseur d'identités peut éventuellement se porter garant de l'acteur auprès d'autres fournisseurs d'identités ou de service, et ce en leur communicant des certificats ou jetons d'authentification. Quant au fournisseur de services, il est comme un consommateur de l'identité de l'acteur qui est issue (directement ou indirectement) du fournisseur d'identités de ce dernier. En d'autres mots, le fournisseur de services est le portail qui protège l'accès à une ressource protégée en vérifiant l'authenticité de l'identité du demandeur d'accès. Trois conceptions d'architecture sont possibles pour une interaction dans laquelle le fournisseur d'identités peut être perçu comme étant un proxy entre l'acteur et le fournisseur de services [32] : *Interactive, Active Client et Credential-based*.

Comme son nom l'indique, un système *Interactive* est basé sur l'interaction entre le fournisseur d'identités et le fournisseur de services. Tel qu'illustré dans la figure 4.4, après que le fournisseur d'identités reçoive la demande d'approbation de la part du fournisseur de services concernant la requête de l'acteur en question, ce dernier doit (au moins une fois) s'authentifier auprès de son fournisseur d'identités.

La configuration *Active Client* (figure. 4.5) est similaire à l'*Interactive* à la différence que la gestion du transfert des messages (jetons, requêtes, attributs, *etc.*) entre le fournisseur de services et d'identité est centrée sur l'acteur. Cela signifie qu'il n'y a pas d'interaction directe entre les deux fournisseurs pour l'authentification de l'acteur. Néanmoins, la délivrance d'un jeton d'authentification de la part du fournisseur d'identités se fait à la connaissance du fournisseur service en question, ce qui signifie qu'à l'image du système *Interactive*, le fournisseur d'identités garde une traçabilité des interactions de l'acteur.

Enfin, la dernière conception *Credential-based* est différente des deux précédentes. En effet, l'acteur s'authentifie indépendamment auprès de son fournisseur d'identités (en fonction du protocole utilisé par ce dernier) et récupère ainsi des *Credentials* qu'il pourra utiliser auprès de fournisseur(s) de service(s) conventionné(s) avec son fournisseur d'identités. Des *Credentials* sont des artefacts

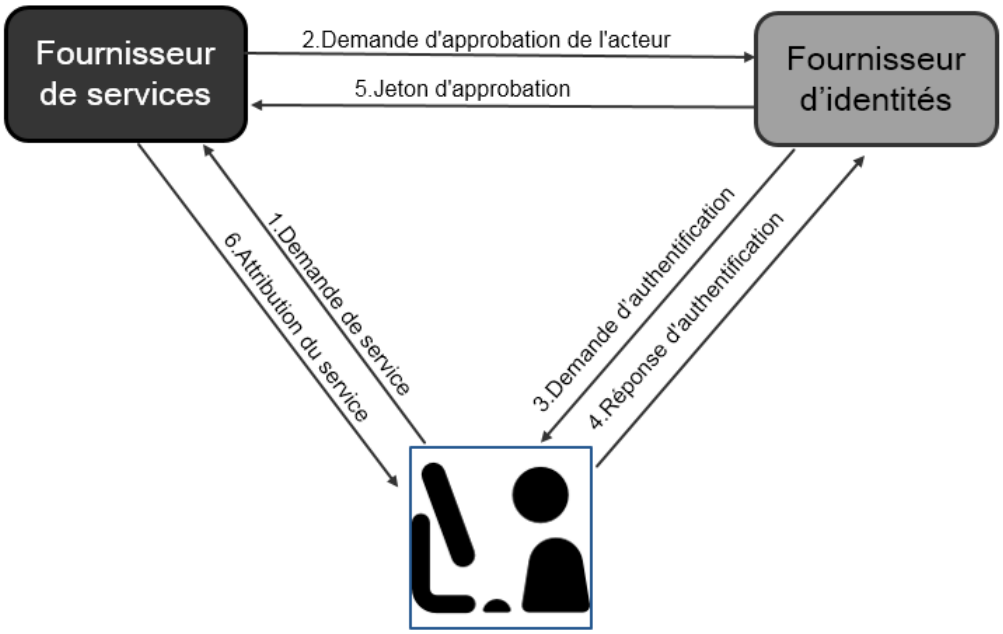

FIGURE 4.4 – Authentification *Interactive*.

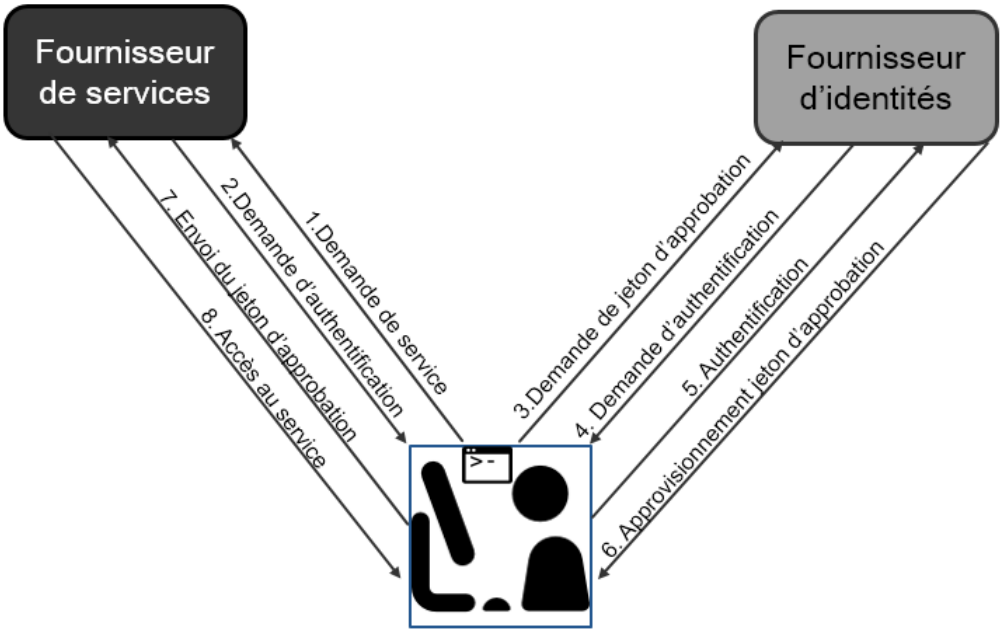

FIGURE 4.5 – Authentification *Active-client*.

transférables fournis par un fournisseur d'identités à un utilisateur authentifié en guise de preuve d'authentification, *e.g.*, signatures numériques, certificats X.509, assertions SAML. Par conséquent, les interactions de l'acteur ne sont plus traçables par le fournisseur d'identités. En outre, l'acteur dispose d'une liberté d'utilisation et de réutilisation des Credentials (figure. 4.6).

Les trois précédentes configurations présentent certains avantages en matière de gestion d'au-

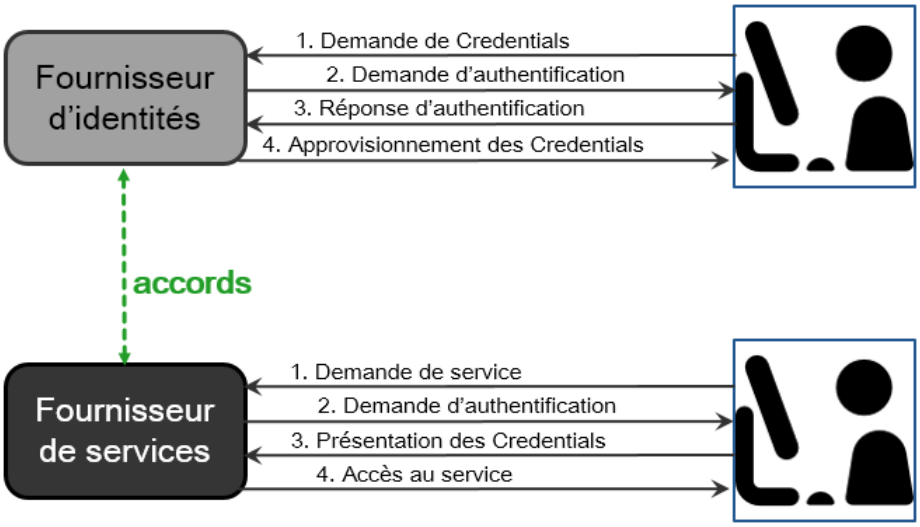

FIGURE 4.6 – Authentification *Credential-Based*.

thentification. En revanche, utilisées telles quelles dans notre contexte de communautés hétérogènes, elles ne sont pas sans failles ni limites. Le système *Interactive* ainsi que *Active-Client* sont assez vulnérables en matière de préservation de vie privée en ce qui concerne les interactions de l'acteur, ce qui donne la possibilité (dans le cas où le fournisseur d'identités est compromis) par exemple de reconstruire des processus métiers décomposés et déployés dans la même, voire dans plusieurs communautés [15]. Par ailleurs, les configurations *Active-Client* et *Credentials-based* qui bannissent toutes interactions directes entre le fournisseur de services et le fournisseur d'identités, concentrent beaucoup de pouvoir sur l'acteur. En fait, le client peut être considéré comme la partie la plus vulnérable par rapport aux fournisseurs d'identités et de services. Ainsi, si le client se fait usurper son identité le fournisseur de services ne pourra pas détecter cette violation. En effet, ce problème concerne davantage la configuration *Active-Client* que la *Credentials-based*. Car cette dernière se base sur un système de combinaison d'attributs permettant de vérifier (au niveau du fournisseur de services) si l'acteur qui présente les *Credentials* est bien celui qu'il prétend être. Cette vérification se fait au moyen d'attribut(s) supplémentaire(s) lié(s) aux Credentials et stocké(s) au niveau du fournisseur de services. Ces attributs font office de clé secrète partagée entre les deux parties, comme proposé par *Idemix* ou *U-Prove*. Néanmoins, sachant que dans notre contexte le fournisseur d'identités peut se faire remplacer par l'entreprise, il est difficile de partager mutuellement de tels attributs entre toutes les entreprises appartenant à une communauté donnée. Cela signifie que l'interaction entre le fournisseur de services et le fournisseur d'identités est inévitable, le challenge étant alors de savoir comment l'exploiter de manière optimale.

Un fournisseur d'identités peut être en parallèle fournisseur de services, dans le sens où il vérifie les identités d'acteurs externes et fournit des accréditations et approbations aux acteurs internes auprès d'autres fournisseurs d'identités et de services. Techniquement, la communication entre deux fournisseurs se fait au moyen de jeton, certificat ou *Credentials* qui dépendent du protocole utilisé au niveau du fournisseur d'identités. Ce moyen de communication doit être compatible avec ce-

lui utilisé au niveau du fournisseur de services. Dans le cas où les fournisseurs d'identités-services utilisés par deux ou plusieurs entreprises d'une même communauté ne sont pas compatibles, la procédure d'authentification peut engendrer des problèmes d'interopérabilité. Pour résoudre ce problème d'interopérabilité dans un contexte fédéré, nous nous sommes basés sur l'utilisation de l'outil *LemonLDAP-NG*.

4.2.3.2 LemonLDAP-NG

LemonLDAP-NG est un outil Open source d'authentification unique. Capable de gérer plus 200 000 utilisateurs [130, 129], LLDAP-NG supporte plusieurs protocoles d'authentification, à savoir : *Active Directory, Apache (Kerberos, NTLM, OTP, etc.), BrowserID (Mozilla Persona), CAS, Facebook (OAuth2.0), Google, LDAP directory, OpenID, Radius, Remote LemonLDAP-NG, Proxy LemonLDAP-NG, SAML 2.0/ Shibboleth, Slave, SSL, Twitter (OAuth), WebID, Yubikey*. Pour le processus d'authentification et la gestion des mots de passe utilisateurs, en plus des protocoles LDAP et Active Directory, LemonLDAP-NG est aussi capable d'utiliser plusieurs types de base de données via une interface Perl (i.e., DBI), à savoir, MySQL, PostgreSQL, Oracle, *etc*. LemonLDAP-NG agit également en tant que fournisseur d'identités via les protocoles SAML / Shibboleth identity-provider, OpenID identity-provider, CAS identity-provider [48]. Les principaux modules qui composent LemonLDAP-NG sont :

- **Manager** : Utilisé pour administrer la configuration de l'utilisation de LemonLDAP-NG et pour parcourir les sessions des utilisateurs.

- **Portal** : Principalement utilisé comme interface d'authentification pour les utilisateurs, le *Portal* propose les différents modes d'authentification disponibles sous LemonLDAP-NG [49]. En outre, le *Portal* fournit les identités numériques grâce au fournisseur d'identités CAS, OpenID et SAML. Le *Portal* agit également en tant que Proxy, permettant de transférer des jetons d'authentification entre mécanismes d'authentification hétérogènes comme OpenID, CAS, *etc*. Par ailleurs, le *Portal* permet de réaliser d'autres fonctionnalités de gestion comme changer le mot de passe, notifier l'utilisateur, afficher une liste des applications autorisées.

- **Handler** : Module Apache utilisé pour protéger les applications au moment où un utilisateur essaye d'y accéder, le *Handler* protège les applications en vérifiant l'existence d'une certification (issue d'un mécanisme d'authentification) de l'identité de l'utilisateur. Si une telle certification n'est pas en possession de l'utilisateur, le *Handler* le redirige vers le *Portal* d'authentification.

La figure 4.7 montre le scénario d'accès à une ressource protégée grâce à LemonLDAP-NG.

48. Tous les fournisseurs d'identités peuvent être utilisés simultanément, *i.e.* sur le même domaine.
49. Les modes d'authentification doivent être préconfigurés sur LemonLDAP-NG installé au niveau du domaine en question.

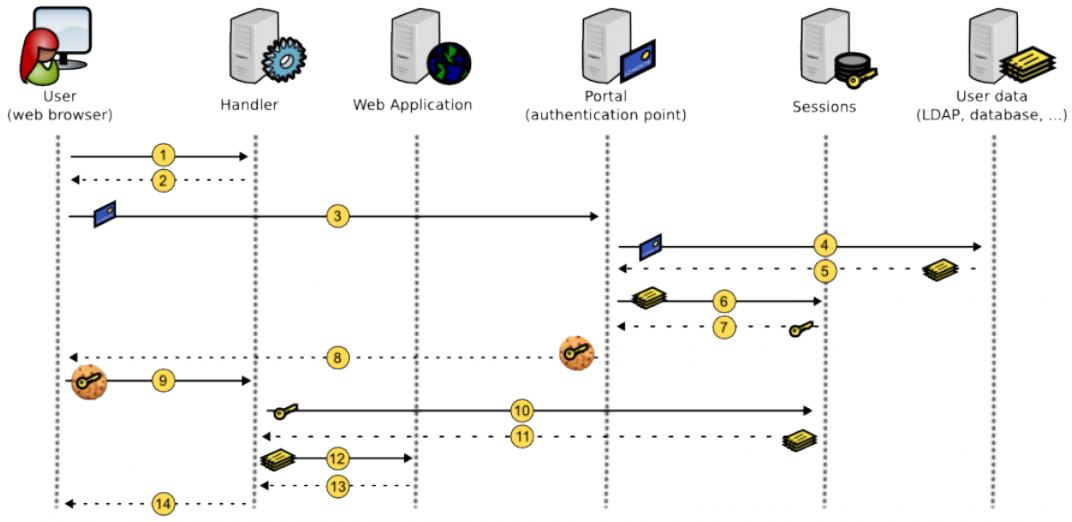

FIGURE 4.7 – *LemonLDAP-NG* mode de fonctionnement [ref].

1. Un utilisateur tente d'accéder à une application protégée, sa requête est interceptée par le *Handler*.
2. Si le *Handler* ne détecte pas de cookie authentification unique (SSO), alors il redirige l'utilisateur au *Portal* pour qu'il s'authentifie.
3. L'utilisateur s'authentifie à travers le *Portal*.
4. Le *Portal* vérifie l'authentification.
5. Si l'utilisateur s'authentifie avec succès, le *Portal* collecte les informations sur l'utilisateur auprès de la base LDAP.
6. Le *Portal* crée ensuite une session pour sauvegarder les informations sur l'utilisateur.
7. Le *Portal* récupère la clé de session.
8. le *Portal* crée un cookie d'authentification unique SSO avec la clé de la session récupérée lors de l'étape précédente.
9. L'utilisateur est redirigé vers l'application protégée muni de son cookie.
10. Le *Handler* récupère la session depuis le cookie fourni par l'utilisateur.
11. Le *Handler* sauvegarde sur son cache les données de l'utilisateur.
12. Le *Handler* communique les informations concernant l'utilisateur en question à l'application protégée en vue de vérifier les droits d'accès.
13. L'application envoie la réponse au *Handler*.
14. Le *Handler* transmet la réponse de l'application à l'utilisateur.

Les protocoles d'authentification dans LemonLDAP-NG peuvent être choisis par l'utilisateur grâce au module *backend choice*. Le module *backend multi* permet de chaîner les authentifications parmi une liste, dans le sens où, dès qu'une authentification réussit, l'utilisateur est connecté. Par ailleurs,

LemonLDAP-NG est adapté aux contextes distribués fédérés tel qu'un RSE grâce à son mode d'authentification *Remote LemonLDAP-NG authentication*. Ce mode de fonctionnement permet une authentification inter-domaines dans le cadre d'une fédération. La *Remote authentication* permet d'utiliser les informations de session d'un utilisateur authentifié dans un domaine donné auprès d'autres domaines extérieurs compatibles. La *Remote authentication* est une forme d'authentification unique car elle permet à un utilisateur de s'authentifier une seule fois pour plusieurs domaines. Ce mode d'authentification fonctionne sous la condition que le *Portal* secondaire [50] soit déclaré au niveau du *Manager* du Portal initial [51] (délégué). Une autre forme d'authentification inter-domaines est la *Proxy authentification*. En effet, LemonLDAP est capable d'agir en tant que Proxy en transmettant des jetons d'authentification uniques d'un portail à un autre. Ce mode d'authentification est intéressant quand il s'agit d'exposer le Portal LemonLDAP pour une authentification interne [52] aux fournisseurs d'identité externes.

Accessoirement, il existe un outil permettant la transformation et la synchronisation de bases de données sous le format LDAP. Cet outil open-source est connu sous le nom de *Ldap Synchronization Connector (LSC)*. *LSC* synchronise les données depuis n'importe quelle source de données, à savoir, une base de données, un annuaire LDAP, ou un simple fichier, tout en lisant, transformant et comparant les données entre les sources et les référentiels de destination [132].

4.2.3.3 Authentification et interopérabilité

LemonLDAP-NG supporte la plupart des protocoles de gestion des identités numériques, à savoir : *Login/PSW, OpenID, OAuth, SSL X509, CAS*, et cela en tant que fournisseur et consommateur d'identité. Par conséquent, nous proposons d'utiliser LemonLDAP-NG comme une *"passerelle"* entre différents mécanismes d'authentification existants. Techniquement, nous utilisons *LemonLDAP-NG* en tant que client de gestion d'authentification au niveau des communautés collaboratives du RSE, et ce en se basant sur les mécanismes d'authentification des entreprises comme fournisseurs d'identité numériques.

Du coté entreprise, nous proposons de mettre en place *LemonLDAP-NG* sur une surcouche logicielle de gestion des identités collaboratives. *LemonLDAP-NG* fera office de fournisseur de jetons d'authentification : *SAML*, *OpenID* et *CAS* auprès des acteurs de l'entreprise. Parallèlement il agira en tant que consommateur d'identité (*i.e.* fournisseur de services) vis-à-vis des communautés collaboratives dans lesquelles sont impliqués les acteurs de l'entreprise. La procédure de consommation a pour but la validation de l'identité de chaque acteur externe. Elle se déclenche suite à la réception d'un jeton de demande d'approbation d'un acteur de la part du gestionnaire d'authentification de la communauté en question.

En effet, pour la conception de notre architecture d'authentification distribuée et fédérée nous nous sommes basés sur une extension de la conception *Credentials-based*. Ce choix permet de préserver la confidentialité des expériences collaboratives de l'acteur afin d'éviter une éventuelle traçabilité

50. Le Portal du domaine de l'application sollicitée.
51. Qui fournit les informations de session de l'utilisateur en question.
52. La base d'information sur les utilisateurs est disponible en interne.

de la part de tiers malveillants. En outre, la conception *Credentials-based* permet de réduire la charge de traitement pour le fournisseur d'identités vu qu'il ignore l'objet de chaque requête et se contente seulement de fournir des *Credentials* universels permettant d'identifier l'acteur en question.

Pour qu'un acteur puisse s'authentifier, il demande dans un premier temps des *Credentials* à son fournisseur d'identités (*i.e.,* son entreprise). Les *Credentials* seront générés en fonction du protocole d'authentification du fournisseur d'identités, *e.g.*, format *OpenID*, ou *SAML*. Ensuite, l'acteur fournit les *Credentials* au gestionnaire d'authentification de la communauté qu'il souhaite rejoindre. Ce dernier, étant compatible (grâce à *LemonLDAP*) avec tous les formats de *Credentials* fournis, sera en mesure de procéder à une procédure d'authentification en vue de la consommation d'une ressource collaborative au sein de la communauté. Pour cela, une solution classique et typique *Credentials-based* consisterait à partager une clé secrète entre le fournisseur d'identités de l'entreprise de l'acteur et le gestionnaire d'identité de la communauté. Comme nous l'avons présenté précédemment, cette démarche n'est pas très sécurisée, ni facile à gérer dans un contexte ouvert tel qu'une communauté RSE. Par conséquent, nous avons modifié la configuration *Credentials-based* (illustrée dans la figure 4.6) en y ajoutant une interaction entre les deux parties d'authentification afin de vérifier que l'identité de l'acteur en question n'a pas été usurpée tout en préservant la confidentialité des interactions collaboratives de l'acteur. La figure 4.8 montre le déroulement de notre processus d'authentification fédérée.

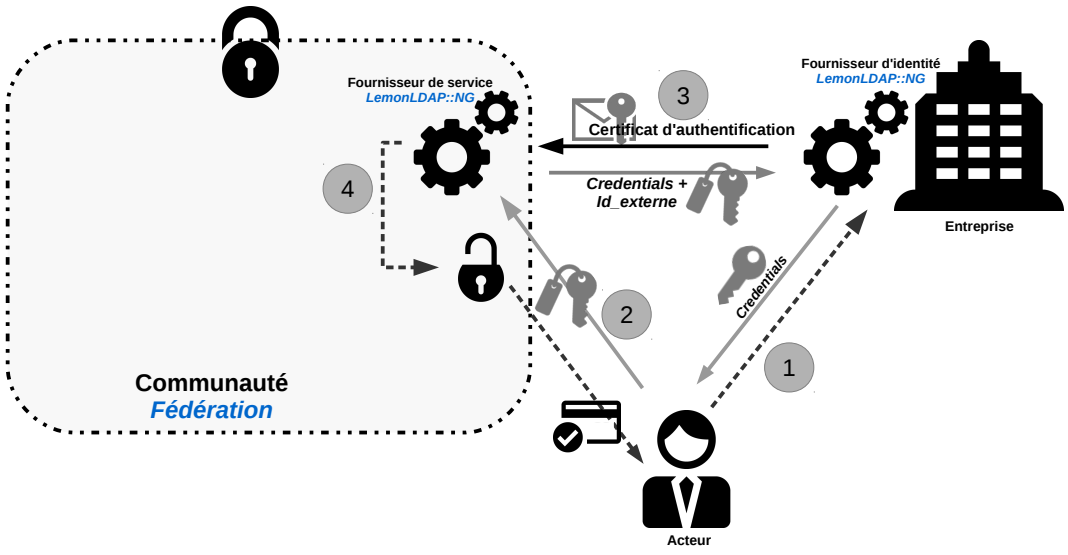

FIGURE 4.8 – Processus d'authentification.

Jusqu'à maintenant, nous avons discuté les deux premières étapes (**1** et **2** de la figure 4.8) du processus d'authentification. La troisième étape illustre l'interaction entre les deux parties. En effet, grâce à notre procédure de gestion des identités des acteurs dans les communautés, nous pouvons

vérifier l'authenticité d'un propriétaire de *Credentials* sans avoir besoin de procédures cryptographiques avancées telles que le partage de clés privées. Plus précisément, notre gestion d'identité est basée sur les notions d'identifiant interne immuable et propre à l'entreprise de l'acteur et un identifiant externe généré par le gestionnaire d'identité de l'entreprise et lié à l'identifiant interne pour une utilisation externe au sein d'une communauté donnée. La génération des *Credentials* est également dépendante de l'identifiant interne. Ainsi, ce mix d'attributs permet de vérifier au niveau du fournisseur d'identités si l'identifiant externe de l'acteur ainsi que les *Credentials* qu'il présente sont conformes, et ce par l'intermédiaire de l'identifiant interne de l'acteur. Plusieurs solutions peuvent être proposées dans le cadre de cette procédure. Une solution très simple serait d'associer une clé unique à chaque identifiant interne et la décomposer en deux parties qui seront associées séparément à chaque instance d'identifiant externe et *Credentials*. Une fois les *Credentials* validés par le fournisseur d'identités de l'entreprise, l'acteur sera authentifié auprès du gestionnaire d'identité de la communauté en question.

Cependant, le processus ne s'arrête pas à cette dernière étape, car l'acteur voudra accéder à des ressources déployées dans le cadre de la communauté mais appartenant à d'autres entreprises. Par conséquent, il est primordial que l'identité de l'acteur soit validée auprès du mécanisme d'authentification de l'entreprise propriétaire de la ressource collaborative désirée. Pour cela, nous nous servons de la confiance mutuelle établie dans le cadre de la fédération. Plus précisément, étant déjà authentifié auprès du mécanisme d'authentification de la communauté, l'identité de l'acteur sera approuvée auprès du mécanisme d'authentification de l'entreprise propriétaire par le service *LemonLDAP* de la communauté. Techniquement, l'approbation se fait au moyen de la communication d'un jeton [53] d'authentification de la part de *LemonLDAP* de la communauté au mécanisme d'authentification choisi par l'entreprise [54]. Cette conception est très avantageuse, car elle permet, d'un côté d'éviter déjà de nombreuses procédures d'authentification vis-à-vis de chaque entreprise appartenant à la communauté ; et d'un autre côté, elle permet également de résoudre le problème d'interopérabilité entre les différents mécanismes d'authentification des entreprises partenaires.

Pour résumer, comme illustré dans la figure Fig 4.9 [55], le gestionnaire d'authentification d'une communauté vérifie l'identité d'un acteur donné en interrogeant le gestionnaire d'identité de son entreprise au moyen des *Credentials* d'identité fournis par le sujet de l'authentification. Une fois authentifié, l'identité de l'acteur sera approuvée auprès de toutes les entreprises appartenant à la communauté (fédération) par le gestionnaire d'identité de la communauté au moyen d'un échange de jetons compatibles (*i.e. LemonLDAP*).

LemonLDAP-NG ne vise pas à remplacer le mécanisme de base de gestion d'identité de l'entreprise. Il sert plutôt à offrir une ouverture à plus de diversité et une flexibilité concernant l'utilisation de mécanismes d'authentification (éventuellement incompatibles) entre les entreprises collaboratives.

53. Le jeton prend la forme d'un cookie.
54. Sur la base des différents protocoles d'authentification proposés par *LemonLDAP*.
55. Les numéros 1,2,3 montre l'ordre des étapes du processus.

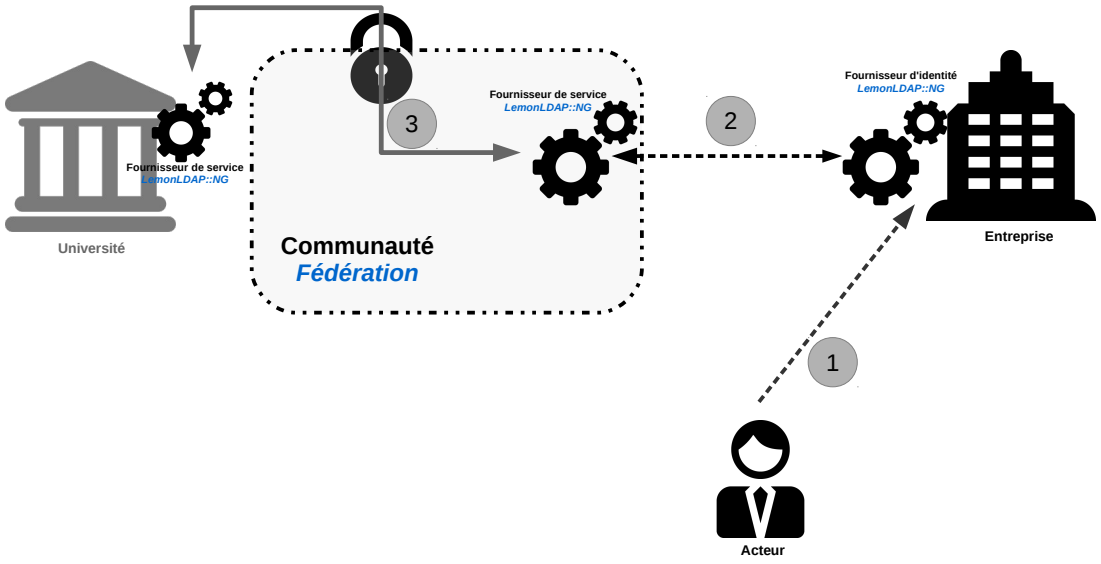

FIGURE 4.9 – Authentification fédérée.

4.3 Conclusion

En résumé, nous avons proposé une conception d'un modèle de collaboration (sous forme de communauté) basé sur une architecture de plateforme *hybride*, *i.e.* un compromis entre une architecture centralisée et décentralisée. Cette architecture nous a permis de répondre aux principaux besoins d'un réseau social d'entreprise à savoir, l'interopérabilité, la sécurisation de données et le passage à l'échelle.

Concernant la gestion des ressources, nous avons proposé que ces dernières restent hébergées au niveau des serveurs des entreprises et que le partage (déploiement) passe d'abord par une phase de virtualisation. À propos de la gestion des identités numériques, notre solution est fondée sur deux notions d'identifiant, à savoir : identifiant interne et identifiant externe. Le dernier est généré sur la base du premier et utilisé au niveau des communautés de collaboration en tant que référence d'attributs identitaire de l'acteur en question. Quant à l'identifiant interne, il est unique et protégé par l'entreprise de l'acteur en question. Le couple de ces identifiants permet une authentification fédérée basée sur une extension de configuration *Credential-based* et l'outil *LemonLDAP-NG*. Ce dernier permet par ailleurs de gérer l'interopérabilité entre les différents mécanismes de gestion d'authentification utilisés par les entreprises au sein de la fédération de collaboration.

L'authenticité des attributs identitaires des acteurs est assurée grâce à la confiance mutuelle qui lie les entreprises dans le cadre d'une fédération. Malgré cela, dans un contexte large et ouvert à grande échelle à plusieurs entreprises, il existe toujours un risque de faire confiance à toutes les entreprises. Pour cela il est nécessaire de cloisonner les cercles collaboratifs entre entreprises même au sein d'une même communauté. Cela justifie davantage la raison pour laquelle nous ne pouvons

pas nous fier uniquement à l'authentification, et met par ailleurs en évidence le besoin d'une phase de contrôle d'accès aux ressources collaboratives. Dans le chapitre suivant, nous aborderons ce contrôle d'accès dans les communautés du RSE OpenPaaS à travers nos contributions sur cet aspect.

Chapitre 5

Contrôle d'accès

Sommaire

5.1 Introduction

Ayant présenté la première partie concernant l'architecture de collaboration ainsi que la gestion des ressources et identités numériques, nous aborderons dans ce chapitre la gestion du contrôle d'accès aux ressources partagées au sein des communautés de collaboration *OpenPaaS* RSE.

5.2 Représentation abstraite du modèle de contrôle d'accès

Avant toute chose, nous tenons à rappeler brièvement les principaux besoins et problèmes liés à la gestion des autorisations dans les communautés *OpenPaaS* RSE. D'abord, nous avons le partage *user-centric* **(cf. section Contrôle d'accès, chapitre Problématique et motivations)** auquel

est associée la problématique de la vérification automatique de la consistance et la cohérence des politiques de contrôle d'accès. Cela nous oriente vers un choix de langage formel de politiques. Le partage user-centric donne aux acteurs le pouvoir de définir des permissions sur des ressources appartenant à leurs entreprises. Ainsi, il faut que les entreprises aient leur autonomie pour garder le contrôle sur leurs ressources partagées. Par ailleurs, le modèle de règles nécessite une grande flexibilité dans un environnement hétérogène tel que le RSE, car les attributs identitaires par rapport auxquels les politiques sont définis (le rôle par exemple) peuvent être hétérogènes et parfois avec une sémantique non-symétrique entre deux ou plusieurs entreprise. En outre, nous avons également souligné l'importance d'avoir la possibilité de définir des autorisations négatives (interdiction) pour permettre à un propriétaire d'une ressource partagée (acteur/entreprise) de s'opposer (suspendre) à une autorisation d'accès à la ressource en question. Ceci implique en outre la nécessité d'envisager des stratégies de combinaison entre des règles et des politiques. D'un autre côté, nous avons présenté et mis en évidence les avantages de la prise en compte du contexte, qui est davantage fructueuse dans le cas où le contexte est exploité d'une manière dynamique et en temps réel afin d'adapter le comportement du mécanisme de décision. Pour cela, nous avons convenu qu'il est plus judicieux d'opter pour une approche événementielle (*i.e.*, basée sur les événements). Pour finir, nous avons discuté la possibilité d'avoir des autorisations particulières, comme la délégation qui consiste en une permission temporaire. Par conséquent, notre choix du langage de politique a été guidé par la précédente contrainte d'un formalisme *"logique"* ajouté au besoin d'inclure le temps pour la gestion des permissions. Notre choix s'est porté sur la logique temporelle *Event-calculus* qui est un langage formel basé sur une logique de premier ordre, et par ailleurs doté d'un raisonneur automatique *"Dec-Reasoner"* (cf. section *Vers une implantation formelle de XACML*, chapitre *État de l'art*). Le modèle de contrôle d'accès que nous proposons dans le cadre de cette thèse tient compte de l'intégralité de ces besoins.

5.2.1 Attribute Based Access Control

Par rapport aux nécessités que nous avons relevées dans le contexte du RSE, ABAC nous permet de répondre aux besoins suivants :

- **Flexibilité du modèle de règle** : car grâce à ABAC, nous avons la possibilité de définir des attributs supplémentaires et les combiner sous la même règle, *e.g.* le grade, les compétences, l'adresse IP, la réputation, le type de terminal, *etc.*.
- **Possibilité de définir des interdictions** : par le biais de l'ajout de l'attribut *type*, qui prend comme valeur *"Permit"* pour une règle d'autorisation, ou *"Deny"* pour une règle d'interdiction.
- **Autonomie des entreprises** : puisqu'elles peuvent définir des contraintes supplémentaires voire même des règles/politiques entières complémentaires à celles définies par les acteurs.
- **Considération du contexte** : grâce à la flexibilité des règles par rapport aux attributs utilisés pour leurs définitions, entre autres le temps, ce qui permet également de définir des permissions éphémères, *i.e.*, délégation.

Un de nos objectifs en matière de contrôle d'accès dans *OpenPaaS* RSE est la possibilité de définir des autorisations temporaires. Par conséquent, nous avons fait le choix d'implanter le modèle

ABAC grâce à un langage formel qui inclut la gestion du temps, à savoir la logique *Event-calculus*. Dans la section suivante, nous présentons notre formalisme du modèle de contrôle d'accès basé sur *Event-calculus*. Cependant, afin d'élucider le déroulement du processus du contrôle d'accès, nous présenterons d'abord l'architecture de contrôle d'accès inspirée de XACML.

5.2.2 Vision d'architecture

Notre vision du RSE est principalement fondée sur le concept de communauté. La figure. 5.1 montre l'architecture de notre framework de contrôle d'accès dans chaque communauté du RSE. Cette architecture est inspirée de l'architecture basique de XACML (*cf. chapitre* ***État de l'art***). Néanmoins, nous l'avons étendu (par rapport à nos besoin *OpenPaaS*) avec un autre module de gestion de la confiance.

1. **Community Enforcement Service (CES)** : ce composant est l'interface entre l'acteur et le système de contrôle d'accès de la communauté. Il sert en outre de pont entre le reste des composants du framework.
2. **Community Information Store (CIS)** : la base d'information de la communauté où les attributs des ressources, acteurs et les organisations sont enregistrées.
3. **Community Administration Store (CAdS)** : la base principale dans laquelle sont ajoutées, administrées et stockées les politiques et règles de contrôle d'accès de la communauté.
4. **Community Decision Service (CDS)** : ce module est responsable de la prise de décision finale vis-à-vis de toute requête reçue. Plus précisément, à la réception d'une requête, le *CDS* recherche les politiques et règles du *CAdS* correspondantes au sujet, génère par la suite les patrons *Event-calculus* et enfin met en application les politiques vis-à-vis de la requête en question au moyen du raisonneur *Dec-Reasoner*.
5. **Community Trust Service (CTS)** : ce service gère la réputation et la confiance numérique des acteurs de la communauté. Pour évaluer et mettre à jour la réputation d'un acteur donné, le *CTS* interagit d'un côté avec le *CES* pour récupérer les informations de session courante de l'acteur en question, et de l'autre côté interagit avec le *CIS* pour récupérer les traces d'interactions historiques de l'acteur.

Les différentes étapes illustrées dans la figure 5.1 peuvent être divisées en deux processus parallèles et complémentaires, à savoir : le processus principal de contrôle d'accès (Bloc A) et un processus complémentaire d'évaluation de la confiance numérique (Bloc B).

Trois modules sont impliqués dans le processus d'évaluation de la confiance, à savoir : le **CES**, le **CTS** et le **CIS**. Le module principal dans ce processus est bien le **CTS**. Ce dernier interagit en premier lieu (étape 1*) avec le **CES** pour récupérer les informations sur le comportement de l'acteur pendant sa session courante. Ensuite (étape 2*), il récupère des informations sur l'historique d'évaluation de la confiance de l'acteur en guise de paramètres pour la procédure d'évaluation de la confiance et la réputation courante du même acteur.

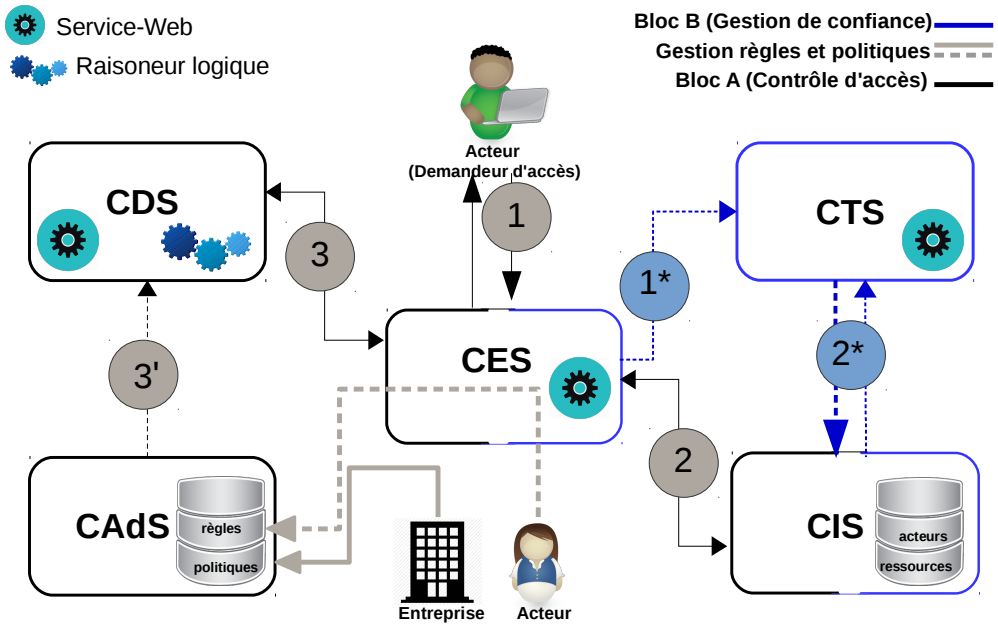

FIGURE 5.1 – Architecture globale du frame-work de contrôle d'accès OpenPaaS.

Parallèlement, le processus du contrôle d'accès inclut quatre modules du framework, à savoir : **CES**, **CIS**, **CDS** et **CAdS**. Le **CAdS** est le conteneur des politiques de sécurité, le noyau autour duquel tout le framework est fondé. Il s'agit d'une base de données administrée par les entreprises ainsi que les acteurs à travers l'insertion, la modification et la suppression de règles et politiques de contrôle d'accès. Le composant principal dans le processus de prise de décision est le **CDS**. Afin qu'il puisse prendre une décision d'autorisation d'accès (Étape 3), le **CDS** interagit directement avec le **CAdS** (étape 3') pour récupérer les règles et politiques par rapport aux attributs d'une requête reçue, qui lui sont transmis par le **CES**. Le **CES** peut récupérer des métadonnées (contextuelles par exemple) sur le sujet, l'objet et la requête avant de les transmettre au **CDS** (étape 2). Une fois la décision prise par le **CDS** et transmise au **CES**, ce dernier se charge de notifier l'acteur de la décision en lui autorisant ou refusant l'accès à la ressource demandée.

5.3 Représentation formelle du modèle de contrôle d'accès

Dans cette partie, nous présenterons en détail tous les composants de notre modélisation du contrôle d'accès dans les communautés *OpenPaaS*, et ce à travers les différents patrons Event-calculus que nous avons proposés.

5.3.1 Modélisation des règles de contrôle d'accès

Dans notre modèle, la règle est le noyau de toutes politiques de contrôle d'accès. Suivant le modèle ABAC, une règle est généralement composée de *Cible (Target)*, *Condition* et *Type*. Le résultat de l'exécution d'une règle est appelé *Effet* qui prend la valeur *Permit* (autorisé) ou *Deny* (non-autorisé).

5.3.1.1 Cible

La cible d'une règle spécifie les attributs utilisés pour vérifier si la règle en question correspond à une requête donnée. Nous proposons une spécification générique de la cible d'une règle en formatant les attributs inclus sous forme de paire *nom-valeur*. Ce choix nous permet d'avoir une grande flexibilité dans la définition des règles. Par exemple, comme attribut de cible on peut avoir, **Nom** : *Ressource*, **Valeur** : *DocumentA*. Dans notre modélisation, la conception de la cible d'une règle inclut : la *ressource* désirée, le *sujet* consommateur, et l'*action* demandée par le sujet sur la ressource. Il est également possible d'offrir une représentation *abstraite* des composants de la cible. Par exemple, l'attribut *"rôle/équipe/organisation"* au lieu de l'attribut *Sujet*, l'attribut *"Vue"* [111] à la place de *"ressource"*, *etc.*. Par ailleurs, une requête est principalement composée des attributs suivants : *sujet*, *objet* et *action*. Nous présenterons en premier lieu notre modèle de vérification de la cible d'une règle par rapport à une requête ($Model1$).

Modèle 1 (Modélisation de la partie Target d'une règle)

sort r, atname, atvalue
predicate AtHasValue (atname, atvalue) **...(1)**
event Match(r), Mismatch(r)
fluent RuleTargetHolds(r)

$\forall$ *r, time* : *Initiates (Match(r), RuleTargetHolds(r), time).* **...(2)**
$\forall$ *r, time* : *Terminates (Mismatch(r), RuleTargetHolds(r), time).* **...(3)**

$\forall$ *r,* : $\neg$ *HoldsAt(RuleTargetHolds(r),0).* **...(4)**

Le **modèle 1** illustre un patron générique de modélisation de la cible d'une règle. Dans ce modèle, nous avons d'abord défini quelques variables, à savoir : *r*, *atname* et *atvalue*. La variable *r* est utilisée pour représenter les règles. Les variables *atname* et *atvalue* sont utilisées pour représenter respectivement le nom d'un attribut et sa valeur. Ensuite, nous avons défini le prédicat **(1)** *AtHasValue* qui permet de spécifier les attributs de la cible. En outre, nous avons défini les événements, *Match* et *Mismatch*, et leur effet *RuleTargetHolds*. Pour gérer les événements, nous avons défini le prédicat **(2)** (*Initiate*) pour l'activation, et le prédicat **(3)** (*Terminates*) pour la désactivation. Le fluent *RuleTargetHolds* sera utilisé comme étant le *but* recherché, qui n'est pas atteint par défaut grâce au prédicat **(4)** (*i.e.*, initialisation à l'instant 0 à faux). En effet, l'événement *Match* se produit quand la cible d'une instance de règle correspond aux valeurs d'attributs de la requête en question.

5.3.1.2 Effet et condition

Une fois la cible vérifiée, la règle peut être évaluée à base des *conditions* et du *type*. Le type de la règle est un attribut indiquant s'il s'agit d'une *Permission* ou d'une *Interdiction*. Quant aux conditions, grâce à notre choix d'un modèle ABAC et au formalisme *Event-calculus*, nous avons la possibilité de considérer plusieurs types de contraintes fonctionnelles et non fonctionnelles. Cependant, nous avons généralisé la modélisation des contraintes sous le volet de la gestion du *contexte*. Cela nous permet de définir des politiques au niveau des entreprises davantage abstraites que celles définies au niveau des acteurs. Plus de détails sur l'aspect de contraintes contextuelles seront présentés dans le chapitre ***Gestion du risque***.

5.3.1.3 Modèle de règle

Dans le **modèle 2**, nous présentons le patron générique de modélisation de règles. Afin de vérifier si une instance de règle s'applique sur la requête, le modèle de règle se base sur la validation de la cible. Pour cela, le **modèle 1** est réutilisé. Si aucune cible de règle ne s'applique à la requête, le résultat obtenu à partir du modèle de règle sera *Not-Applicable*. Sinon, la règle autorise l'accès (*RulePermitted*), ou le refuse (*RuleDenied*).

Modèle 2 (Modèle de règle de contrôle d'accès)

sort r, s
fluent [RulePermitted/RuleDenied/RuleNotApplicable](r)
event [ApproveRule/DisApproveRule/RejectRule](r)

$\forall$ *r, time* : *Initiates (ApproveRule(r), RulePermitted(r), time).*
$\forall$ *r, time* : *Initiates (DisApproveRule(r), RuleDenied(r), time).*
$\forall$ *r, time* : *Initiates (RejectRule(r), RuleNotApplicable(r), time).*

$\forall$ *r, time, s* : *Happens (ApproveRule(r),time)* $\rightarrow$
HoldsAt (TargetHolds(r),time) $\wedge$ *HoldsAt (ContextHolds(s),time)* $\wedge$
HoldsAt (RuleTypePermission(r),time). **...(1)**

$\forall$ *r, time, s* : *Happens (DisApproveRule(r),time)* $\rightarrow$
HoldsAt(TargetHolds(r),time) $\wedge$ *[*$\neg$ *HoldsAt(ContextHolds(s),time)* $\vee$
$\neg$ *HoldsAt(RuleTypeProhibition(r),time)].* **...(2)**

$\forall$ *r, time* : *Happens (RejectRule(r),time)* $\rightarrow$ *HoldsAt (TargetDoesntHold(r),time).*
$\forall$ *r* : $\neg$ *HoldsAt (RulePermitted(r),0).* **...(3)**
$\forall$ *r* : $\neg$ *HoldsAt (RuleDenied(r),0).* **...3**
$\forall$ *r* : $\neg$ *HoldsAt (RuleNotApplicable(r),0).* **...3**

Dans ce dernier patron nous avons défini une variable ***s*** qui représente le sujet de la règle. Les principaux fluents sont utilisés pour indiquer si l'effet de la règle sera, *Permit*, *Deny* ou *Not-Applicable*. Les événements de contrôle des états des fluents sont *ApproveRule*, *DisApprouveRule* et *RuleDoesNotApply*, pour signifier respectivement la que requête est approuvée, rejetée ou la règle n'est même pas applicable sur les attributs de la requête. Pour cela, nous nous sommes basés sur le prédicat *Initiate* pour gérer ces trois événements.

L'axiome **(1)** contrôle l'occurrence de l'événement *ApproveRule* à travers plusieurs conditions. En premier lieu, la cible de l'instance de la règle en question doit correspondre aux attributs de la requête reçue. Ensuite, le contexte au moment de la réception de la requête ne doit pas présenter des exceptions de sécurité. Cet aspect sera détaillé plus tard dans le cadre des règles entreprise (**modèle 11**). Cependant, afin que la gestion des règles reste simple pour des utilisateurs non expérimentés, le contexte est ignoré dans le cadre des règles définies par les acteurs et considéré comme étant validé pour toutes les requêtes. Enfin, le type de la règle doit être une *permission* et non une *interdiction*. Ces trois dernières conditions satisfaites, l'événement (*ApproveRule*) peut ainsi se produire et changer par conséquent l'état du fluent *RuleIsPermitted*, autorisant par la suite le sujet de la requête en question. Toujours avec des opérateurs logiques, l'axiome **(2)** réalise le processus inverse pour vérifier si la requête reçue doit être rejetée. Enfin, les prédicats **(3)** initialisent touts les effets de toutes règles à *faux*.

5.3.1.4 Instance et évaluation de règle

Le **modèle 3** illustre un exemple d'instance d'une règle particulière, à savoir la règle *"RuleJames"*, mettant en évidence les valeurs des attributs de la cible.

Modèle 3 (Instance de règle)

fluent RuleTypePermission(r)
fluent RuleTypeProhibition(r)
r RuleJames
atname Subject, Object, Action

$\forall\ r, time\ :\ Happens(Match(r),\ time) \wedge AtHasValue(Subject,\ atvalue_1) \wedge AtHasValue(Object,\ atvalue_2) \wedge AtHasValue(Action,\ atvalue_3) \rightarrow atvalue_1 = James \wedge atvalue_2 = CV\_Jessy.pdf \wedge atvalue_3 = Read.$ **(...1)**

$\forall\ r, time\ :\ Happens(Mismatch(r),\ time) \wedge AtHasValue(Subject,\ atvalue_1) \wedge AtHasValue(Object,\ atvalue_2) \wedge AtHasValue(Action,\ atvalue_3) \rightarrow atvalue_1 \neq James \wedge atvalue_2 \neq CV\_Jessy.pdf \wedge atvalue_3 \neq Read.$ **(...2)**

HoldsAt(RuleTypePermission(RuleBob),0). **(...3)**
$\neg$ *HoldsAt(RuleTypeProhibition(RuleBob),0).* **...3.**

La règle *RuleJames* est définie par *Jessy* en vue du partage de son CV en lecture avec *James*. Les attributs de cette règle sont *Subject=James, Object=CV_Jessy.pdf, Action=Read*. Ainsi, si le sujet *James* établit une requête de demande de lecture à la ressource CV_Jessy.pdf, la cible sera validée (axiome (**1**)), sinon cette règle ne sera pas applicable sur la requête de *James* (axiome **2**). Les prédicats (**3**) servent à définir le type de la règle, en l'occurrence *permission* pour *RuleJames*.

Les objectifs (du raisonnement par abduction) sont illustrés dans le **modèle 4**. Ces objectifs concernent en premier lieu la vérification de la cible (**1**), ensuite la vérification de l'effet de règle *i.e., Permit/Deny* (**2**).

Les résultats du raisonnement sur l'instance de règle *RuleJames* sont illustrés dans ***Solution 1*** et ***Solution 2***. La ***Solution 1*** montre la vérification de la cible, tandis que la ***Solution 2*** montre le

résultat complet du raisonnement sur la requête par rapport à la règle.

Modèle 4 (Objectifs)

$\forall\ r\ :\ HoldsAt(RuleTargetHolds(r),\ 1) \vee \neg\ HoldsAt(RuleTargetHolds(r),\ 1).$ **(...1)**

$\forall\ r\ :\ HoldsAt(RuleIsPermitted(r),\ 2) \vee HoldsAt(RuleIsDenied(r),\ 2) \vee HoldsAt(RuleIsNotApplicable(r),\ 2).$ **(...2)**

Les résultats du raisonnement sur la partie de vérification de la cible par le raisonneur *Dec-Reasoner* sont illustrés dans *Solution1*.

Solution 1 (Évaluation de la Target d'une règle par Dec-Reasoner)

t_0
Happens(Match(RuleJames), 0).
t_1
+RuleTargetHolds(RuleJames).

La ***solution 1*** montre qu'à l'instant $\mathbf{t}_1$ le fluent indiquant la correspondance de la requête avec la cible de la règle *RuleJames* est validé. Cette correspondance a été vérifiée à l'instant $\mathbf{t}_0$ (par rapport aux prédicats **(1)** et **(2)** du ***modèle 3***) provoquant l'occurrence de l'événement *Match* pour l'instance *RuleJames*.

La solution 2 montre que, à partir de l'instant $\mathbf{t}_1$, les fluents relatifs au contexte et la cible sont déjà vérifiés et que le *type* de la règle est une *permission*. Ainsi, l'événement *ApproveRule* peut se produire et changer l'état du fluent *RuleIsPermitted* à *Vrai*, après qu'il a été initialisé à *Faux* à l'instant $\mathbf{t}_0$ (***modèle 2*** » prédicat **(3,...)**). Dans le cas où l'un des fluents défini dans l'axiome de contrôle de l'événement de la règle n'est pas validé, alors c'est le fluent *RuleIsDenied* qui change d'état à *Vrai* (***modèle 3*** » prédicat **(2)**). Autrement, dans le cas où c'est la cible qui ne correspond pas, le résultat de retour sera *RuleNotApplicable*, *i.e.,* la conséquence de l'événement *RejectRule* dans le ***modèle 2***.

Solution 2 (Résultats d'évaluation de règle par Dec-Reasoner)

t_0
+ ContextHolds(James).
RuleTypePermission(RuleJames).
t_1
+ TargetHolds(RuleJames).
Happens(ApproveRule(RuleJames), 0).
t_2
+ RulePermitted(RuleJames).

5.3.2 Modélisation de politiques de contrôle d'accès

Une politique est un ensemble de règles. Cela signifie que le modèle de gestion de politique est principalement basé sur des stratégies de combinaison de règles. Sachant qu'une règle peut avoir comme effet *Permit*, *Deny* ou *Not-applicable*, le modèle de politiques doit pouvoir prendre une décision vis-à-vis d'une requête sur la base des effets des règles qui composent la politique en question. Ainsi, comme une règle, l'effet d'une politique prend comme valeurs *Deny*, *Permit* ou *NotApplicable*.

Les stratégies de combinaison que nous avons utilisées dans nos modèles de contrôle d'accès sont *Deny-takes-precedence*, *Permit-takes-precedence*, *All-Deny*, *All-Permit*, *All-NotApplicable*. Cependant, ces stratégies doivent être utilisées selon une certaine discipline. Par conséquent, afin de former une politique, nous avons procédé avec deux types de regroupement de règles, l'un basé sur tous les attributs de la cible d'une requête, et l'autre basé uniquement sur le sujet de cette dernière.

5.3.2.1 Patron de politiques basées sur la cible

La première méthode de définition d'une politique est de faire un regroupement de règles en se basant sur la même cible contenue dans la requête reçue, à savoir : le sujet, l'objet et l'action. Par conséquent, la politique contient uniquement des règles qui sont applicables sur la requête reçue. Cette méthode de regroupement par cible est utilisée dans le cas où une ressource appartient à au moins deux ou plusieurs entités collaboratrices. Il suffit qu'une seule entité revendique l'accès à la ressource pour que la requête soit rejetée. Cependant, si tous les propriétaires de la ressource sont d'accord pour accorder l'accès, la requête sera autorisée. Par conséquent, dans cette méthode de formation de politique, nous opérons les stratégies de combinaison : *All-permit* et *Deny-takes-precedence*.

Le patron générique de politiques basées sur la cible est illustré dans le **modèle 5**, dans lequel les politiques sont référencées par la variable **p**. Les stratégies de combinaison sont définies à travers les axiomes **(1)** pour *Deny-takes-precedence* et **(2)** pour *All-permit*. Dans l'axiome **(1)**, pour chaque politique ***p***, règle ***r*** et instant *t*, il suffit qu'une seule règle appartenant à la politique soit à effet *Deny* pour que l'événement *DisApprovePolicy* se produise à l'égard de cette politique, provoquant ainsi un effet *Deny* pour la politique à travers le fluent *PolicyDenied*. Inversement, l'axiome **(1)** vérifie que toutes les règles d'une politique autorisent l'accès à la ressource demandée, *i.e.*, effet *Permit*.

Modèle 5 (Modèle de politique : regroupement par cible)

sort p, r
fluent PolicyPermitted(p) fluent PolicyDenied(p) fluent PolicyNotApplicable(p)
event ApprovePolicy(p) event DisApprovePolicy(p) event RejectPolicy(p)
predicate PolicyHasRules(p,r)

$\forall$ *p, time* : *Initiates (ApprovePolicy(p), PolicyPermitted(p), time).*
$\forall$ *p, time* : *Initiates (DisApprovePolicy(p), PolicyDenied(p), time).*
$\forall$ *p, time* : *Initiates (RejectPolicy(p), PolicyNotApplicable(p), time).*

#Combination strategy : Deny-takes-precedence
$\forall$ *p, r, time* : *Happens(DisApprovePolicy(p),time)* $\rightarrow \exists$ *r* : *PolicyHasRules(p,r)* $\wedge$ *HoldsAt(RuleDenied(r),time).* **...(1)**

#Combination strategy : All-permit
$\forall$ *p, r, time* : *Happens(ApprovePolicy(p),time)* $\wedge$ *PolicyHasRules(p,r)* $\rightarrow$ *HoldsAt(RulePermitted(r),time).* **...(2)**

$\forall$ *p* : $\neg$ *HoldsAt (PolicyPermitted(p),0).*
$\forall$ *p* : $\neg$ *HoldsAt (PolicyDenied(p),0).*

5.3.2.2 Patron de politiques basées sur le sujet

Dans cette méthode, les politiques sont formées à base des règles concernant le même sujet, à savoir le sujet de la requête reçue. Par conséquent, on peut trouver dans la politique des règles qui ne sont pas applicables sur la requête en question. Cette manière de définir des politiques consiste à trouver au moins une règle dans l'ensemble permettant d'accréditer ou non le sujet en question. Cette méthode se base sur l'hypothèse qu'il ne peut pas y avoir deux règles avec les mêmes cibles et de types différents. En revanche, si aucune règle n'est applicable sur la ressource en question avec l'action désirée, l'effet de la politique sera *Not-Applicable*. Par conséquent, nous procédons au moyen des stratégies *(permit/deny)-takes-precedence* avec *All-NotApplicable*. Ces stratégies de combinaison sont ainsi choisies pour ce genre de politiques afin d'éviter les éventuels conflits de combinaison. En effet, si les stratégies *Deny-overrides*, *All-permit* avec *All-NotApplicable* sont utilisées ensemble, on aura une incohérence dans le cas où une règle qui satisfait la cible est permise, et que dans la politique il existe certaines règles qui ne sont pas applicables.

La modélisation de ce type de politiques basées sur le sujet est illustrée dans le **modèle 6**. Les stratégies *Deny-takes-precedence* et *Permit-takes-precedence* sont mises en place respectivement à travers les axiomes **(1)** et **(2)** de préconditions de l'événement *DisApprovePolicy* et *ApprovePolicy*. Quant à l'axiome **(3)**, il vérifie si toutes les règles de la politique ne s'appliquent pas à la cible de la requête reçue afin d'ignorer la politique en rendant l'effet *PolicyNot-applicable*.

Modèle 6 (Modèle de politique : regroupement par sujet)

sort p, r
fluent PolicyPermitted(p) fluent PolicyDenied(p) fluent PolicyNotApplicable(p)
event ApprovePolicy(p) event DisApprovePolicy(p) event RejectPolicy(p)
predicate PolicyHasRules(p,r)

$\forall$ *p, time* : *Initiates (ApprovePolicy(p), PolicyPermitted(p), time).*
$\forall$ *p, time* : *Initiates (DisApprovePolicy(p), PolicyDenied(p), time).*
$\forall$ *p, time* : *Initiates (RejectPolicy(p), PolicyNotApplicable(p), time).*

 # Combination strategy : Deny-takes-precedence :
$\forall$ *p, r, time* : *Happens(DisApprovePolicy(p), time)* $\rightarrow \exists$ *r* : *PolicyHasRules(p,r)* $\wedge$ *HoldsAt(RuleDenied(r), time).* **...(1)**

#Combination strategy : Permit-takes-precedence :
$\forall$ *p, r, time* : *Happens(ApprovePolicy(p), time)* $\rightarrow$
$\exists$ *r* : *PolicyHasRules(p,r)* $\wedge$ *HoldsAt(RulePermitted(r),t).* **...(2)**

#Combination strategy : All-NotApplicable :
$\forall$ *p, r, time* : *Happens(RejectPolicy(p), time)* $\wedge$ *PolicyHasRules(p,r)* $\rightarrow$ *HoldsAt(RuleNotApplicable(r), time).* **...(3)**
$\forall$ *p* : $\neg$ *HoldsAt(PolicyPermitted(p),0).*
$\forall$ *p* : $\neg$ *HoldsAt(PolicyDenied(p),0).*
$\forall$ *p* : $\neg$ *HoldsAt (PolicyNotApplicable(p),0).*

À la fin des deux modèles de politiques, nous initialisons les fluents correspondant aux différents effets de la politique, à savoir *Permited*, *Denied* ou *NotApplicable*.

5.3.2.3 Instance et évaluation de politique

Pour définir une instance de politique, nous avons utilisé le prédicat *PolicyHasRule* qui spécifie quelles règles rentrent sous la sphère de quelle politique. Ainsi, nous avons défini (conformément à l'exemple de motivation) la politique concernant la candidature de *Jessy*, qui doit envoyer son *CV* à *James* pour *lecture*. Nous avons appelé cette politique *JessyApplication*. Nous avons conçu cette politique avec les deux méthodes de combinaison, à savoir : cible et sujet. Le **modèle 7** montre une instance de politique basée sur des règles ayant la même cible (*Sujet/Objet/Action*) que la requête reçue, à savoir : *James/CV_Jessy.pdf/Read*.

Modèle 7 (Policy (Target) specification)

policy JessyApplication

PolicyHasRule(JessyApplication, RuleJessy).
PolicyHasRule(JessyApplication, RuleBob).
PolicyHasRule(JessyApplication, RuleAlice).

Cette politique est composée de trois règles, *RuleJessy*, *RuleBob*, *RuleAlice* qui sont définies respectivement par : *Jessy*, *Bob* et *Alice*. Dans leurs règles, *Jessy* et *Alice* autorisent *James* à lire le *CV* de *Jessy*, *i.e.,* les deux règles sont de types *Permission*. Cependant, *Bob*, l'encadrant de *Jessy*, revendique l'accès (pour une raison donnée) via sa règle qui est de type *Interdiction*. Quand on invoque le raisonneur pour ce patron, il donne les résultats qui apparaissent dans la ***solution 3***.

Solution 3 (Résultats d'évaluation de politique (Target) par Dec-Reasoner)

$\mathbf{t}_0$
...
Happens(ApproveRule(RuleJessy), 0).
Happens(ApproveRule(RuleAlice), 0).
Happens(DisApproveRule(RuleBob), 0).
$\mathbf{t}_1$
\+ *RulePermitted(RuleJessy).*
\+ *RulePermitted(RuleAlice).*
\+ *RuleDenied(RuleBob).*
Happens(DisApprovePolicy(JessyApplication), 1).
$\mathbf{t}_2$
\+ *PolicyDenied(JessyApplication).*

Le ***Modèle 8*** illustre une instance de politique basée sur le même sujet. Dans l'ensemble des règles qui composent cette politique, seule la règle définie par *Jessy* s'applique sur la requête reçue, et approuve par ailleurs cette demande d'accès. Le résultat du *Dec-Reasoner* figure dans la ***solution 4***. Il est à noter que dans la solution décrite par le raisonneur *Dec-Reasoner*, le signe moins (-) précédant un fluent signifie sa valeur est *faux*, le signe (+) signifie que la valeur du fluent est *vrai*.

Modèle 8 (Policy (sujet) specification)

policy JessyApplication

PolicyHasRule(JessyApplication, RuleEve).
PolicyHasRule(JessyApplication, RuleWalter).
PolicyHasRule(JessyApplication, RuleNestor).
PolicyHasRule(JessyApplication, RuleJessy).
PolicyHasRule(JessyApplication, RuleMatilda).

Le résultat du raisonnement est le suivant :

Solution 4 (Résultats d'évaluation de politique (Sujet) par Dec-Reasoner)

$\mathbf{t}_0$
...
Happens(Mismatch(RuleEve), 0).
Happens(Mismatch(RuleWalter), 0).
Happens(Mismatch(RuleNestor), 0).
Happens(Match(RuleJessy), 0).
Happens(Mismatch(RuleMatilda), 0).
$\mathbf{t}_1$
- *RuleTargetHolds(RuleEve).*
- *RuleTargetHolds(RuleWalter).*
- *RuleTargetHolds(RuleNestor).*
+ *RuleTargetHolds(RuleJessy).*
- *RuleTargetHolds(RuleMatilda).*
$\mathbf{t}_2$
Happens(RejectRule(RuleEve), 2).
Happens(RejectRule(RuleWalter), 2).
Happens(RejectRule(RuleNestor), 2).
Happens(ApproveRule(RuleJessy), 2).
Happens(RejectRule(RuleMatilda), 2).
$\mathbf{t}_3$
+ *RuleNotApplicable(RuleEve).*
+ *RuleNotApplicable(RuleWalter).*
+ *RuleNotApplicable(RuleNestor).*
+ *RulePermitted(RuleJessy).*
+ *RuleNotApplicable(RuleMatilda).*
$\mathbf{t}_4$
Happens(ApprovePolicy(JessyApplication), 4).
$\mathbf{t}_5$
+ *PolicyPermitted(JessyApplication).*

5.3.3 PolicySet

À partir de cette section, nous appellerons les règles et politiques définies par les sujets *règle/politique "de partage"*.

Un *policySet* est un ensemble qui permet de regrouper plusieurs politiques. En *XACML*, le policySet est considéré comme la racine de toute autorisation d'accès. Dans notre contexte RSE, nous avons exploité le concept de policySet pour mettre en parallèle les politiques de partage de ressources et les politiques des entreprises. Ainsi, une entreprise garde le contrôle à travers ses politiques sur

les politiques de partage de ressources, et ce au moyen de la combinaison de ces deux niveaux de politiques.

En effet, les stratégies de combinaisons prennent en compte uniquement les effets (*Permit/Deny*) de politiques, ce qui signifie que les politiques de partage et les politiques d'entreprises sont indépendantes. Cette indépendance permet aux entreprises d'avoir la possibilité de définir des politiques avec un niveau d'abstraction plus élevé et générique. Cependant, afin de pouvoir combiner des stratégies de contrôle d'accès, les deux parties (entreprise, sujet) doivent se mettre au même niveau sur la structure XACML. Par exemple, en respectant la structure XACML, on ne peut pas combiner des règles avec des politiques.

Donc, si l'entreprise décide d'avoir un niveau d'abstraction plus élevé de sa stratégie de contrôle d'accès, elle doit s'aligner avec les stratégies de contrôle d'accès définies par les sujets au niveau des politiques. En effet, à l'image d'une politique de partage, une politique d'entreprise peut être composée de plusieurs règles. En revanche, afin de respecter l'abstraction recherchée dans les politiques d'entreprise, cette dernière ne peut pas être basée sur la combinaison d'instances de règles comme celles définies par les sujets. Une politique d'entreprise doit être définie sur la base de paramètres plus génériques qui ne se focalisent pas sur chaque instance de partage de ressources. Cela permet aux entreprises d'avoir une approche flexible de contrôle d'accès.

Modèle 9 (Modèle de PolicySet)

sort pS, p
fluent PolicySetPermitted(pS), PolicySetDenied(pS), PolicySetNotApplicable(pS)
event ApprovePolicySet(pS) event DisApprovePolicySet(pS) event RejectPolicySet(pS)
predicate PolicySetHasPolicy(pS,p)

$\forall$ *pS, time* : *Initiates (ApprovePolicySet(pS), PolicySetPermitted(pS), time).*
$\forall$ *pS, time* : *Initiates (DisApprovePolicySet(pS), PolicySetDenied(pS), time).*
$\forall$ *pS, time* : *Initiates (pejectPolicySet(pS), PolicySetNotApplicable(pS), time).*

#Combination strategy : Deny-takes-precedence
$\forall$ *pS, p, time* : *Happens(DisApprovePolicySet(pS),time)* $\rightarrow \exists$ *r* : *PolicySetHasPolicy(pS,p)* $\wedge$ *HoldsAt(PolicyDenied(p),time).* **...(1)**

$\forall$ *pS, p, time* : *Happens(ApprovePolicySet(pS),time)* $\wedge$ *PolicySetHasPolicy(pS,p)* $\rightarrow$ *HoldsAt(pulePermitted(p),time).* **...(2)**

$\forall$ *p* : $\neg$ *HoldsAt (policySetPermitted(pS),0).*
$\forall$ *p* : $\neg$ *HoldsAt (policySetDenied(pS),0).*

Le ***modèle 9*** illustre le patron de modélisation d'un policySet. Comme c'est décrit dans les prédicats **(2)** et **(3)**, ce modèle se base sur des stratégies de combinaison de politiques, à savoir des politiques de partage ainsi que des politiques d'entreprise.

5.3.3.1 Politique d'entreprise

Pour définir les politiques d'entreprises, nous nous sommes basés sur une approche d'*évaluation du risque*, dont de plus amples détails seront présentés dans le chapitre **cf. Évaluation du risque**.

Dans cette partie, nous nous contentons uniquement de présenter les patrons de vérification des politiques d'entreprise basées sur le risque. L'idée principale du mode opératoire du contrôle d'accès de l'entreprise via le mécanisme de l'évaluation du risque est basé sur la comparaison d'une *valeur de risque* qui résulte d'un ensemble de paramètres avec un *seuil* de tolérance de risque.

Les paramètres sur lesquels est basée la métrique d'évaluation du risque sont la confiance du sujet de la requête, la vulnérabilité du contexte et l'impact d'un éventuel accès non autorisé sur la ressource en question. La métrique de calcul de la valeur du risque ainsi que le choix de paramètres seront détaillés dans le chapitre **cf. Évaluation du risque**. Quant au seuil du risque, c'est le paramètre essentiel défini par l'entreprise pour vérifier si la menace reflétée par une requête donnée, approuvée par une politique de partage existante, peut être tolérée ou pas.

Le ***modèle 10*** illustre la conception d'une politique d'entreprise, où ces nouveaux paramètres liés au risque sont intégrés.

Modèle 10 (Politique d'entreprise)

sort s, p, userTrLevel, orgTHR
predicate UserTrlevel(userTrLevel)
predicate ResourceImpact(impact)
predicate Vulnerability(vul)
predicate OrganizationRiskThold(orgTHR)

fluent PolicyPermitted(p) , PolicyDenied(p) , ContextHolds(s)

event ApprovePolicy(p) , DisApprovePolicy(p) , RiskHolds(s) , RiskDoesNotHold(s)

$\forall$ *p, time* : *Initiates (ApprovePolicy(p), PolicyPermitted(p), time).* **...(1)**
$\forall$ *p, time* : *Initiates (DisApprovePolicy(p), PolicyDenied(p), time).* **...(1')**

$\forall$ *s, time* : *Initiates (RiskHolds(s), ContextHolds(s), time).* **...(2)**
$\forall$ *s, time* : *Terminates (RiskDoesNotHold(s), ContextHolds(s), time).* **...(2')**

$\forall$ *s, time, userTrLevel, impact, vul, orgTHR* : *Happens (RiskHolds(s),time)* $\wedge$ *UserTrlevel(userTrLevel)* $\wedge$ *ResourceImpact(impact)* $\wedge$ *Vulnerability(vul)* $\wedge$ *OrganizationRiskThold(orgTHR)* $\rightarrow$ *orgTHR* $\geq$ *((1-userTrLevel)+vul+impact)/3.* **...(3)**

$\forall$ *s, time, userTrLevel, impact, vul, orgTHR* : *Happens (RiskDoesNotHold(s),time)* $\wedge$ *UserTrlevel(userTrLevel)* $\wedge$ *OrganizationRiskThold(orgTHR)* $\wedge$ *ResourceImpact(impact)* $\wedge$ *Vulnerability(vul)* $\rightarrow$ *orgTHR* $<$ *((1-userTrLevel)+vul+impact)/3.* **...(4)**

$\forall$ *s, p, time* : *Happens (ApprovePolicy(p),time)* $\rightarrow$ *HoldsAt(ContextHolds(s),time).* **...(5)**
$\forall$ *s, p, time* : *Happens (DisApprovePolicy(p),time)* $\rightarrow \neg$ *HoldsAt(ContextHolds(s),time).* **...(5')**

$\forall$ *s* : $\neg$ *HoldsAt (PolicyPermitted(s),0).*
$\forall$ *s* : $\neg$ *HoldsAt (PolicyDenied(s),0).*
$\forall$ *s* : $\neg$ *HoldsAt (ContextHolds(s),0).*

Dans ce modèle de politique d'entreprise, les variables *userTrLevel*, *orgTHR*, *impact* et *vul* représentent respectivement la réputation (confiance) du sujet de la requête, le seuil de tolérance de l'entreprise par rapport au risque d'une requête, l'impact provoqué dans l'éventualité où la ressource demandée est compromise et la vulnérabilité du contexte d'accès. Le noyau d'une politique

d'entreprise est la métrique d'évaluation du risque définie à travers les axiomes **(3)** et **(4)**. Ainsi, les événements essentiels pour ce modèle de politiques sont *RiskHolds* et *RiskDoesNotHold* qui sont en l'occurrence contrôlés par des conditions liées aux paramètres que nous avons définis. Ces deux événements sont gérés grâce aux prédicats **(2)** et **(2')**, qui dépendent de la satisfaction des conditions des axiomes **(3)** et **(4)**. Le prédicat **(2)** permet de changer l'état du contexte en mettant la valeur du fluent *ContextHolds* à *vrai*, ce qui signifie que le contexte est validé pour la requête reçue. En revanche, à l'occurrence de l'événement *RiskDoesNotHold*, le prédicat **(2')** met le contexte dans un état de *non-validation* provoquant ainsi l'interdiction de l'accès réclamé par la requête (axiomes **(5')**, **(1')**). Inversement, l'accès est autorisé pour la requête reçue jugée non-risquée par la politique d'entreprise (axiomes **(5)**, **(1)**).

5.3.4 Synthèse

Cette première partie porte sur le contrôle d'accès dans les environnements RSE. Nous avons commencé par une représentation abstraite du modèle de contrôle d'accès dans laquelle nous avons précisé que nous nous sommes basés sur un modèle ABAC avec une conception d'architecture inspirée d'XACML, et ce afin d'avoir un modèle de contrôle d'accès flexible en matière de définition de règles. Ensuite nous avons abordé la modélisation formelle de notre mécanisme de contrôle d'accès dans laquelle nous avons présenté les différents patrons de conception basés sur la logique temporelle *Event-calculus*, à savoir les règles de partage de ressources, les politiques de partage et d'entreprise et les policySets qui permettent de combiner les deux niveaux de politiques (sujet et entreprise). Dans la section suivante, nous aborderons une particularité du contrôle d'accès liée aux RSEs, à savoir les *délégations* qui consiste en des autorisations temporaires.

5.4 Délégation

La délégation est un type de permission qui permet à un sujet de définir des règles temporaires au profit d'un autre sujet vis-à-vis de certaines ressources. L'avantage de l'utilisation de ce type d'autorisation réside dans le fait qu'une délégation est auto-révocable. Par conséquent, la délégation réduit considérablement l'administration des règles de contrôle d'accès dans des perspectives d'indisponibilité de sujets qui s'occupent des tâches ne devant pas être interrompues. Prenons un exemple simple où l'utilisateur *Bob* doit quitter temporairement sa communauté, et certaines tâches de *Bob* doivent être réalisées. Dans de telles circonstances, l'idéal serait que *Bob* puisse déléguer les tâches qui doivent être réalisées à un autre sujet (un collègue par exemple) jusqu'à son retour. Par ailleurs, supposons que *Bob* soit débordé et qu'une date limite d'une livraison approche. Une bonne solution pour *Bob* consiste à déléguer ses travaux à une autre personne qualifiée, et ce pour un intervalle de temps donné, par exemple la date butoir de la livraison.

5.4.1 Présentation formelle de délégation

La délégation est une forme particulière d'autorisation de contrôle d'accès. Plus précisément, le processus de modélisation du mécanisme est plus au moins similaire à celui du contrôle d'accès, à savoir : la modélisation des cibles, règles, politiques et policySets. En revanche, la spécificité de délégation est liée à l'aspect temporel ainsi qu'à certaines conditions concernant les deux parties de délégation, à savoir : le **délégant** qui délègue l'accès à une ressource et le **déléguataire** qui consomme cette ressource par le biais de l'autorisation déléguée.

L'aspect temporel d'une règle de délégation peut être considéré sous deux formes, une durée basée sur des contraintes liées au délégant, ou bien pour une durée préalablement déterminée. La validité de la première forme de délégation dépend de la disponibilité du délégant. Dans ce cas, une règle de délégation n'est valable que dans le cas où le délégant est hors-ligne, *i.e.*, quitte (temporairement) la communauté de collaboration. Dans le cadre de notre conception d'OpenPaaS RSE, ce type de délégation concerne uniquement les sujets internes d'une entreprise. Quant à la seconde forme de délégation, basée sur une période prédéfinie, elle concerne les sujets externes, à qui certaines tâches d'un acteur interne (qui n'est pas en mesure de les accomplir) seront déléguées.

Une politique de délégation reste similaire à une politique de règle standard. Plus précisément, une politique de délégation est basée sur les mêmes stratégies de combinaison des effets de règles. Cependant, vu que dans une politique de délégation nous nous focalisons principalement sur le sujet déléguataire, nous avons fait le choix que les politiques soient construites sur la base des règles ayant le même *sujet*. Sachant que la modélisation d'une politique de délégation est la même qu'une politique de règles, nous allons dans ce qui suit détailler les deux formes de délégation et présenter uniquement les modèles liés aux règles de délégation.

5.4.1.1 Délégation interne

Comme son nom l'indique, une délégation *"interne"* implique uniquement les sujets internes à une entreprise en tant que déléguataires à travers les autorisations d'un délégant également interne. En d'autres termes, une délégation interne est une forme particulière de partage de droits entre collègues appartenant à une même entreprise. Par exemple, pendant son absence, *James* délègue à sa collègue *Alice* la tâche de gestion de démarches de recrutement des étudiants stagiaires tel que *Jessy* au sein de la communauté collaborative. Cependant, la supervision sur les délégations d'autorisations de la part de l'entreprise ne doit pas être négligée. Vu qu'il s'agit de sujets internes à l'entreprise, pour contrôler les délégations cette dernière se contente uniquement de contraintes supplémentaires s'ajoutant aux règles de délégation définies par les sujets. Car la mise en place des règles entreprises basées sur le risque n'est pas vraiment nécessaire vu que les paramètres d'évaluation du risque (confiance, vulnérabilité environnement et impact de ressources) sont censées être optimaux au sein de la sphère privée de l'entreprise. Selon notre point de vue, les contraintes de délégation internes concernent plus le niveau d'habilitation du sujet déléguataire. Plus précisément, chaque sujet peut réaliser uniquement les tâches que son niveau d'habilitation au sein de son entreprise lui permet de faire, et ce en fonction des politiques internes et subjectives de gestion d'habilitation de son entreprise. Le

principal avantage d'un tel mode de délégation est que la durée de délégation est indéterminée dans le temps, mais contrôlée au moyen de contraintes de disponibilité et d'habilitation. En outre, la délégation interne permet de simplifier la gestion des règles de délégation vu qu'on est pas obligé de redéfinir (mettre à jour) la même règle de délégation chaque fois qu'elle arrive au terme de sa durée prédéfinie.

Modèle 11 (Modèle de règle de délégation)

sort r, s
fluent DelegRulePermitted(r) fluent DelegRuleDenied(r) fluent DelegRuleNotApplicable(r)
fluent DelegRuleInValid(r) fluent IsQualified(s)

event ApproveDelegRule(r) event DisApproveDelegRule(r) event RejectDelegRule(r)

$\forall$ *r, time* : *Initiates (ApproveDelegRule(s), DelegRulePermitted(s), time).*
$\forall$ *r, time* : *Initiates (DisApproveDelegRule(s), DelegRuleDenied(s), time).*
$\forall$ *r, time* : *Initiates (RejectDelegRule(s), DelegRuleNotApplicable(s), time).*

$\forall$ *r, time, s* : *Happens (ApproveDelegRule(r),time)* $\rightarrow$
HoldsAt(TargetHolds(r),time) $\wedge$ *HoldsAt(IsQualified(s),time)* $\wedge$ *HoldsAt(DelegInValid(r),time).* **...(1)**

$\forall$ *r, time, s* : *Happens (DisApproveDelegRule(s),time)* $\rightarrow$
HoldsAt(TargetHolds(s),time) $\vee \neg$ *HoldsAt(IsQualified(r),time)* $\vee$ *HoldsAt(DelegInValid(r),time)*
. **...(2)**

$\forall$ *r, time* : *Happens (RejectDelegRule(r),time)* $\rightarrow$ *HoldsAt (TargetDoesntHold(r),time).*
$\forall$ *r* : $\neg$ *HoldsAt (DelegInValid(r),0).* **...(3)**

Le ***modèle 11*** illustre la modélisation formelle de ce type de règle de délégation interne. Globalement, le modèle est assez similaire à celui des règles classiques à l'exception de deux fluents spécifiques à la délégation, à savoir : le fluent *DelegRuleInValid* et le fluent *IsQualified*. Le fluent *DelegRuleInValid* permet d'indiquer la validité de l'autorisation de délégation en fonction de la disponibilité du délégant. Par ailleurs, quand son état est à *vrai*, le fluent *IsQualified* signifie que le sujet est habilité à accomplir les tâches qui lui sont déléguées. La combinaison des contraintes de disponibilité et d'habilitation se fait grâce à la vérification des prédicats **(1)** et **(2)** pour respectivement autoriser ou rejeter la requête de délégation reçue. La vérification de la contrainte de disponibilité est illustrée sur le ***modèle 13***. Quand à la contrainte d'habilitation, elle est détaillée dans la procédure illustrée sur le ***modèle 12***.

Contrainte d'habilitation

Dans le ***modèle 12***, le niveau d'habilitation d'un sujet *s* ainsi que le seuil minimum d'habilitation sont récupérés par le *CES* au moment de la réception de la requête depuis le *CIS*. Afin de valider le niveau d'habilitation du sujet, l'axiome **(1)** vérifie si la valeur de l'attribut de confiance du déléguataire est supérieure ou égale au seuil défini par l'entreprise propriétaire de la ressource demandée. Inversement, l'axiome **(2)** ne valide pas l'habilitation du sujet déléguataire.

Modèle 12 (Modèle de vérification d'habilitation d'un sujet)

sort s, userQL, orgTHR
predicate UserQualifLevel(userQL) predicate OrganizationThold(orgTHR)
fluent IsQualified(s)
event QualifHolds(s) event QualifDoesNotHold(s)

∀ *s, time* : *Initiates (QualifHolds(s), IsQualified(s), time).*
∀ *s, time* : *Terminates (QualifDoesNotHold(s), IsQualified(s), time).*

∀ *s, time, userQL, orgTHR* : *Happens (QualifHolds(s),time)* ∧ *UserQualifLevel(userQL)* ∧ *OrganizationThold(orgTHR)* → *orgTHR* ≥ *userQL.* **...(1)**

∀ *s, time, userQL, orgTHR* : *Happens (QualifDoesNotHold(s),time)* ∧ *UserQualifLevel(userQL)* ∧ *OrganizationThold(orgTHR)* → *orgTHR* < *userQL.* **...(2)**

∀ *s* : ¬ *HoldsAt (IsQualified(s),0).*

Disponibilité du délégant

Dans le **modèle 13**, nous vérifions la disponibilité d'un sujet délégant représenté par la variable *d*. Pour cela, nous nous basons sur l'occurrence des événements *Connect* et *Disconnect* pour changer l'état du statut du sujet délégant *IsOnLine* entre *vrai* et *faux*. Les prédicats **(1)** et **(2)** permettent de changer l'état de la validité de la règle de délégation, à savoir le fluent *DelegInValid*. Ce dernier fluent est impliqué dans le modèle principal de vérification de la règle de délégation (***Modèle 11***). L'occurrence de l'événement *SuspendDeleg* change l'état du fluent *DelegInValid* à *vrai* et ainsi suspend la validité de la règle. À l'opposé, l'occurrence de *RunDeleg* permet de rétablir la validité de la règle en changeant le fluent *DelegInValid* à *faux*. Quant aux axiomes **(3)** et **(4)**, ils permettent respectivement de (re)suspendre et (re)activer automatiquement l'instance courante de la règle de délégation en fonction de la disponibilité du sujet délégant ***"d"*** en question.

Modèle 13 (*delegator* ***Status Verification model)***

sort d, s

fluent IsOnLine(d) fluent DelegInValid(s)
event Connect(d) event Disconnect(d) event RunDeleg(s) event SuspendDeleg(s)

∀ *d, time* : *Initiates(Connect(d), IsOnLine(d), time).*
∀ *d, time* : *Terminates(Disconnect(d), IsOnLine(d), time).*

∀ *s, time* : *Initiates(SuspendDeleg(s), DelegInValid(s), time).* **(1)**
∀ *s, time* : *Terminates(RunDeleg(s), DelegInValid(s), time).* **(2)**

∀ *d, s, time* : *Happens (SuspendDeleg(s),time)* → *IsOnline(d).* **(3)**
∀ *d, s, time* : *Happens (RunDeleg(s),time)* → ¬ *IsOnline(d).* **(4)**

∀ *d, time* : *Happens (Connect(d),time)* → ¬ *IsOnline(d).*
∀ *d, time* : *Happens (Disconnect(d),time)* → *IsOnline(d).*

5.4.2 Délégation externe

Une deuxième manière de concevoir des règles de délégation vis-à-vis d'un déléguataire externe à l'entreprise consiste à définir des permissions valables uniquement pour un intervalle de temps prédéfini par le délégant. Le **modèle 14** illustre comment une durée de vie d'une règle est établie par rapport à un temps d'expiration prédéfini.

Modèle 14 (Vérification de la durée de la délégation)

sort s, t_{Start}, t_{End}

fluent DelegInValid(s)

$\forall\ s, t_{Start}, t_{End}$: *Trajectory(DelegRulePermitted(s),* t_{Start}*, DelegInValid(s),* t_{End}*).*

Dans le **modèle 14**, nous avons défini deux points de temps, à savoir, $\mathbf{t}_{Start}$ et $\mathbf{t}_{End}$, pour désigner respectivement le temps d'activation et d'expiration de l'autorisation de délégation. Grâce au prédicat *Trajectory*, quand l'état du fluent *DelegRulePermitted* est vrai, ce qui signifie que la règle est autorisée pour la première fois, l'état du fluent *DelegInValid* prend la valeur vraie après un intervalle de temps calculé à base de la somme des deux points de temps de la règle, *i.e.*, $\mathbf{t}_{Start} + \mathbf{t}_{End}$. Par exemple, la durée d'une règle donnée est prédéfinie à 5 unités de temps. Supposons que la règle en question soit activée (autorise l'accès vis-à-vis d'une requête donnée) à l'instant 2. Ainsi, à partir de l'instant 8 ((5+2)+1) la règle ne sera plus valable, ce qui signifie que la même requête préalablement autorisée sera rejetée après l'instant 7. Le fluent *DelegInValid* qui est initialisé à *faux* au niveau du **modèle 11** (axiome **(3)**), et ce pour chaque nouvelle instance de règle de délégation. Tant que l'état de ce fluent reste à *faux*, la règle de délégation reste valide.

5.4.3 Application des modèles sur l'exemple de motivation

Le **modèle 15** illustre l'instance de règle *DelegAlice1* du scénario dans lequel *James* délègue à *Alice* les droits de gestion des recrutements des stagiaires.

Modèle 15 (Instance d'une règle de délégation basée sur la durée de validité)

r DelegAlice1

$\forall$ *s, o, a, d, time* : *Happens(Match(DelegAlice1),time)* $\rightarrow$ *s* $=$ *Alice* $\wedge$ *o=Calendar* $\wedge$ *a=PUT* $\wedge$ *d=James* $\wedge$ *ruleState=Activated* $\neg$ *HoldsAt(DelegInValid(DelegAlice1),time).* **...(1)**

$\forall$ *s, o, a, d, time* : *Happens(MisMatch(DelegAlice1),time)* $\rightarrow$ *s* $\neq$ *Alice* $\vee$ *o* $\neq$ *Calendar* $\vee$ *a*$\neq$ *PUT* $\vee$ *d*$\neq$ *James* $\vee$ *ruleState*$\neq$*Activated* $\vee$ *HoldsAt(DelegInValid(DelegAlice1),time).* **...(2)**

Dans cette instance de règle, des valeurs sont assignées aux attributs de la règle de délégation, à savoir ceux de la cible *(s : Sujet/ o : Objet/ a : Action)* ainsi que le délégant *d*. Avant la phase de raisonnement sur la règle de délégation, les attributs de la requête d'*Alice* seront accompagnés par

l'information concernant l'identité du délégant, en l'occurrence *James*. En outre, la validité de la règle est exprimée grâce au fluent *DelegInValid*, dont l'état doit être à *faux* pour que la règle soit valide. Ainsi, pour que *Alice* puisse consommer ses droits délégués, d'abord il faut que l'ensemble de tous les attributs de sa requête correspondent à ceux de la règle *DelegAlice1*. En plus, la règle *DelegAlice1* doit être valide au moment où la requête a lieu (*axiome* **(1)**). Si un des attributs ne correspond pas ou la validité de la requête arrive à terme, la requête sera par conséquent rejetée (*axiome* **(2)**).

5.5 Conclusion

Dans ce chapitre, nous avons présenté notre contribution concernant l'aspect gestion du contrôle d'accès dans les environnements collaboratifs hétérogènes d'*OpenPaaS* RSE. Nous avons commencé par la représentation abstraite de notre modèle de contrôle d'accès, conçu sur la base d'un modèle ABAC qui répond à nos besoins exprimés dans la problématique. Nous avons par ailleurs illustré et détaillé une vision architecturale de notre framework de contrôle d'accès. Ensuite, nous avons abordé la représentation formelle utilisée pour la mise en œuvre de notre modèle de politique, en l'occurrence une formalisation logique du modèle XACML basée sur la logique temporelle *Event-Calculus*. Enfin, nous avons abordé une particularité du contrôle d'accès liée à notre contexte RSE, à savoir la délégation, qui consiste principalement à associer un contexte temporel à une autorisation d'accès. Nous avons considéré la délégation sous deux différents formes, à savoir les délégations internes et les délégations externes. Nous avons basé notre définition de la délégation interne sur un cloisonnement intra-entreprise et des contraintes de disponibilité du délégant et de niveau d'habilitation du déléguataire. Quant à la délégation externe, elle dépasse les frontières de l'entreprise dans le sens où elle est principalement basée sur une période de validité pré-établie au moment de la définition de l'autorisation. Elle est en outre confrontée à des politiques plus avancées impliquant des politiques de partage simples définies par les sujets et les politiques de l'entreprise qui sont basées sur une évaluation du risque de de la confiance. Dans les deux chapitres suivants, nous aborderons les questions sur la conception détaillée des composants clés d'une politique entreprise, à savoir, la gestion du risque et de la confiance numérique.

Chapitre 6

Gestion du risque

Sommaire

6.1 Introduction

Dans ce chapitre, nous allons présenter notre métrique d'évaluation du risque d'une requête de demande d'accès. Cette métrique est le principal fondement de notre modélisation des politiques entreprises. Nous allons en premier lieu brièvement rappeler le contexte et la motivation qui nous ont orientés vers une conception d'un modèle de contrôle d'accès dynamique basé sur la gestion du risque. Ensuite, nous parlerons du principe d'alignement des concepts standards de gestion du risque avec ceux du contrôle d'accès et les environnements collaboratifs RSE. Nous finirons avec une étude expérimentale qui illustre l'apport du mécanisme de gestion du risque en matière de décision de contrôle d'accès.

6.2 Contexte

Notre principale motivation derrière la volonté d'avoir un mécanisme de contrôle d'accès dynamique pour la gestion des autorisations au sein des communautés OpenPaaS est le partage user-centric, *i.e.*, basé sur le sujet de la collaboration. En effet, nous considérons qu'il est important que les

entreprises puissent garder le contrôle sur leurs ressources partagées par leur personnel (acteurs). Une solution possible est la mise en place de règles entreprise qui font office d'exception pour les règles de partage, et ce pour chaque nouvelle règle, *i.e.*, à chaque partage de ressource. Cependant, comme il s'agit d'un environnement ubiquitaire (fréquence élevée de partage), l'encadrement des règles de partage par l'entreprise ne peut pas se faire à une granularité aussi fine. Le contrôle doit se faire d'une manière plus abstraite via un mécanisme générique qui prend comme paramètres d'entrée les attributs de la cible de la requête de demande d'accès. Ainsi, l'objectif consistera à vérifier si une politique de partage est adaptée à la requête reçue à l'instant de réception de cette dernière. En d'autres termes, cela permet de remettre en cause les politiques utilisateurs qui peuvent être mal définies, obsolètes, voire même non-autorisées (cas d'une usurpation d'identité). Ainsi, la question est de savoir comment traiter les attributs d'une cible d'une requête afin de mesurer les éventuelles menaces et leurs impacts au sein d'une communauté qui ne peut être sans vulnérabilités.

Par ailleurs, le cadre RSE présente une autre spécificité qui est la copropriété de ressources. Car concrètement, dès lors qu'une ressource est partagée, elle devient en réalité la propriété de tous les acteurs qui la partagent. Ainsi, il est important de prendre en considération cette notion de propriété.

Enfin, il est également intéressant de prendre en considération l'hétérogénéité des communautés en matière de gestion des identités et de l'authentification. En effet, dans notre gestion de l'authentification dans OpenPaaS RSE, chaque entreprise peut garder son mécanisme (protocole) d'authentification. Ainsi, vu que les mécanismes d'authentification ne reflètent pas tous la même robustesse en matière de fiabilité dans la certification des identités numériques, cela peut être considéré comme une vulnérabilité du système de contrôle d'accès vu qu'il se base sur des identités authentifiées. Par conséquent, prendre en considération la fiabilité du mécanisme d'authentification peut contrebalancer ces vulnérabilités dues à la volonté de préservation du mécanisme d'authentification pour chaque entreprise.

Dans ce sens, nous proposons d'améliorer les mécanismes de contrôle d'accès classiques grâce une approche basée sur l'évaluation du risque reflété par les requêtes de demande d'accès. En mixant les métriques standards d'évaluation du risque avec les concepts fondamentaux du contrôle d'accès, on peut offrir une solution aux entreprises qui leur permette de rejeter certaines requêtes considérées comme suspectes, ou pas assez fiables. Cela se fait au moyen d'un seuil minimum de tolérance de risque défini par l'entreprise. La métrique d'évaluation du risque, avec le seuil de l'entreprise, constituent les principaux composants des politiques entreprises qui se mixent à celles des utilisateurs au niveau du *PolicySet*.

6.3 Métrique d'évaluation du risque

Le *NIST*[56] définit le risque comme étant la probabilité d'une menace et son impact sur le système [151]. Afin de mener à bien notre approche de formalisation du risque, nous avons d'abord besoin d'aligner les concepts du risque avec ceux du contrôle d'accès et le contexte RSE. Nous consi-

56. National Institue of Standards and Technology : http ://www.nist.gov/

dérons le cas d'un attaquant qui vole des informations sensibles (*i.e.,* **menace**) à travers une interface compromise (*i.e.,* la **vulnérabilité**) qui conduit à des pertes concernant l'image (la réputation) de l'entreprise (*i.e.,* **impact**). Dans ce sens, l'évaluation du risque se résume à la formule suivante ([6], [151]) : $risque = vulnérabilité \times menace \times impact \Longleftrightarrow f(V,M,I)$.

L'**impact** d'un accès non autorisé est étroitement lié à la ressource demandée. Étant donné que certaines ressources sont plus importantes ou sensibles que d'autres, elles nécessitent davantage de vigilance pour leur protection. En outre, certaines actions sur une ressource peuvent aussi avoir des conséquences plus graves que d'autres. En effet, une information très confidentielle ne doit pas être facile d'accès, même en lecture. Dans le même sens, un rapport public peut être facilement consultable (*i.e.,* lecture), néanmoins il ne doit être ni modifié ni effacé par un accès non autorisé. Par conséquent, dans notre approche, nous proposons d'évaluer l'impact du risque en évaluant l'importance de la *ressource* et les conséquences que peut avoir l'*action* désirée sur cette ressource.

Les **vulnérabilités** conduisant à un accès non autorisé dans une communauté RSE sont généralement dues à un mécanisme d'authentification qui peut être trompé (*i.e.*, défaillant). En effet, si l'identité de l'utilisateur effectuant la requête ne peut pas être garantie, il existe un risque que l'utilisateur soit un usurpateur. En réalité, certains mécanismes d'authentification sont plus sûrs que d'autres. Une identification en deux étapes par exemple est plus difficile à tromper qu'une simple connexion par mot de passe. Mais imposer un mécanisme d'authentification fort, et éventuellement coûteux comme l'authentification biométrique, à tous les utilisateurs n'est ni recommandé, ni évident, notamment dans un contexte hétérogène comme le RSE.

Concernant la **menace**, elle émane directement du *sujet* de la requête. En effet, c'est l'action du sujet qui peut générer l'événement et ses conséquences, et ce d'une manière volontaire (*i.e.* attaque), ou involontaire (*i.e.* erreur). Par conséquent, pour évaluer la probabilité d'une attaque, nous proposons d'évaluer la fiabilité d'un utilisateur. Par exemple, le système ne doit pas être aussi indulgent avec les utilisateurs qui essaient souvent d'accéder aux ressources non autorisées, qu'avec ceux qui ont un comportement exemplaire.

Nous rappelons que le principal objectif derrière l'évaluation du risque consiste à trouver une métrique optimale qui permette de définir les politiques abstraites d'entreprise. En effet, nous considérons d'une part que le RSE est très dynamique en matière d'ajout/suspension d'utilisateurs/ressources/communautés, et que d'autre part, une entreprise n'est pas en mesure de connaître en temps réel les informations sur toutes les entités au sein des différentes communautés où ses utilisateurs sont impliqués et ses ressources sont déployées. Par conséquent, nous proposons que l'évaluation du risque se fasse au moyen d'un mécanisme commun et centralisé au niveau de chaque communauté, et que les entreprises définissent un **seuil** de tolérance de risque en dessous duquel toutes les requêtes seront rejetées. C'est de cette manière que, dans notre système, les entreprises gardent le contrôle sur les décisions établies par un mécanisme délégué de contrôle d'accès sur leurs ressources.

6.3.1 Définitions

Dans cette section, nous allons formaliser les concepts précédemment définis, à savoir, *Vulnérabilité*, *Menace* et *Impact*. En outre, nous montrerons comment nous évaluons le risque en utilisant des attributs d'une cible de requête de demande d'accès avec le contexte.

Definition 1 (Impact). *L'impact dépend de l'action demandée sur une ressource donnée à travers la requête reçue. Plus la ressource est importante, plus l'impact sera élevé. Nous considérons qu'une ressource est importante lorsque le nombre d'autorisations définies sur cette ressource est réduit. Par conséquent, pour une requête "req" qui implique l'utilisateur "u", la ressource "r", et l'action "a", nous calculons la moyenne du nombre de règles qui approuvent l'opération d'accès "a" sur la ressource "r" à l'égard de tous les utilisateurs de la communauté en question.*

$$Impact(a,r) = 1 - \frac{\sum_{u \in User} Policy(u,a,r)}{Card(User)} \tag{6.1}$$

Avec

$User$ *l'ensemble des utilisateurs de la communauté,*

$a \in Action$, *l'ensemble des actions (i.e. {R, W, X}),*

$r \in Resource$, *l'ensemble de ressources de la communauté,*

$Policy$, *les décisions de politiques de contrôle d'accès (*$accept$*=1,* $reject$*=0),*

$Card(User)$, *le nombre des utilisateurs de la communauté.*

Nous prenons l'opposé de la moyenne, étant donné que l'impact baisse quand le nombre d'utilisateurs ayant accès à la ressource en question augmente.

Definition 2 (Vulnérabilité). *Une vulnérabilité dépend de la fiabilité du mécanisme d'authentification implanté au niveau de la communauté. En effet, nous considérons que les mécanismes d'authentification existant ne sont pas sans failles de sécurité et par conséquent peuvent être trompés. Par ailleurs, l'efficacité des mécanismes d'authentification dépend du contexte d'utilisation. De plus, à notre connaissance, il n y a pas de standard qui classe les mécanismes d'authentification existant par l'ordre de leur robustesse. Par conséquent, nous considérons le classement des mécanismes d'authentification comme étant un paramètre subjectif qui dépend des termes du contrat de fédération entre les entreprises dans le cadre de la communauté en question. Dans les expérimentations (**section 6.3**) élaborées dans le cadre de cette thèse nous nous sommes basés sur le classement (subjectif) croissant suivant :* $Auth$ *= {Guest, PIN, Login/password, OAuth, 2 Step Validation, Biometric}. Par conséquent, nous attribuons à chaque mécanisme d'authentification un score qui représente le niveau robustesse. Ainsi, pour une requête donnée* req *:*

$$\boxed{Vulnerability(req) = Score(A_C)} \tag{6.2}$$

Avec

C, *la communauté où la requête* req *est établie,*

A_C, *le mécanisme d'authentification* C,

$Score : Auth \rightarrow [0,1]$, *the strength level of* A_C

Notons que les scores sont également subjectifs et sont définis par le créateur de la communauté. Un exemple de score est illustré dans la table 6.1.

Definition 3 (Menace). *La menace dépend de la confiance du sujet à l'origine de la requête, en fonction de l'évaluation de sa réputation sur la base de son historique d'interactions au sein de la communauté. Plus la confiance de l'utilisateur est élevée, moins le risque sera élevé. Comme la valeur de confiance d'un utilisateur appartient à l'intervalle* $]0,1[$*, nous interprétons cela par la formule suivante :*

$$\boxed{Threat(u) = 1 - Trust(u)} \tag{6.3}$$

Avec

$Threat$, *la menace,*

$Trust$, *la confiance numérique du sujet,*

u, *le sujet.*

Avec notre mécanisme d'évaluation de risque, n'importe quelle métrique d'évaluation de confiance peut être facilement intégrée. Cependant, dans le cadre de cette thèse, nous nous basons sur le mécanisme d'évaluation de réputation que nous avons proposé et qui sera présenté dans le chapitre ***Confiance numérique***.

Definition 4 (Risque). *Dans notre contexte, nous avons testé plusieurs formules de combinaison des paramètres précédents. Nous avons fini par choisir une approche linéaire, vu les performances qui en résultent. Ces résultats sont détaillés dans la* ***section (6.3) expérimentation****. En outre, dans certains contextes, il est possible qu'un paramètre soit plus important qu'un autre, c'est pourquoi nous avons utilisé des coefficients pour paramétrer cette importance, sachant que tous les coefficients (ou poids) sont par défaut égaux à* 1.

$$Risk(req) = \frac{k_v \times V(C) + k_t \times T(u) + k_i \times I(a,r)}{k_v + k_t + k_i} \tag{6.4}$$

Avec

V, la vulnérabilité, et son coefficient k_v,

C, la communauté où la requête req est établie,

T, la menace, et son coefficient k_t,

u, le sujet de la requête req,

I, l'impact, et son coefficient k_i,

a, l'action demandée à travers la requête req,

Dans notre cas, nous considérons que l'impact de compromettre une ressource est plus important que les autres paramètres. Ainsi, nous lui avons attribué le poids 3, comparé à 1 pour les autres paramètres. L'utilité de la pondération peut être constatée à partir de la figure 6.1. Dans ce graphique, nous avons fixé la vulnérabilité au niveau maximum : 1.

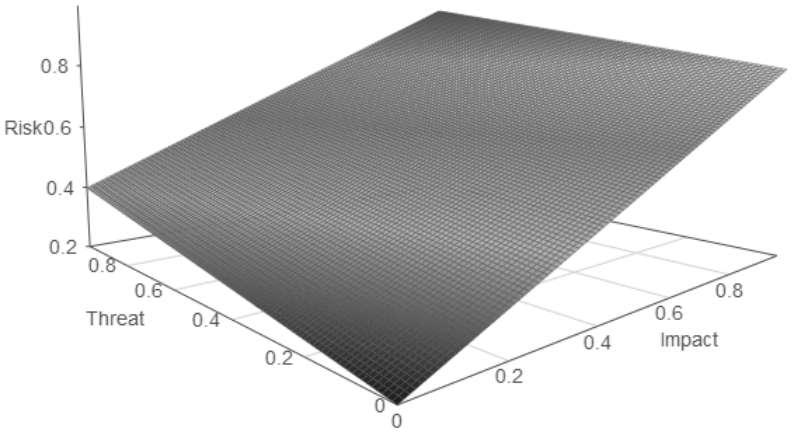

FIGURE 6.1 – Les valeurs du risque pour une pondération {3,1,1} et une vulnérabilité maximale.

Seuil de tolérance du risque

Une entreprise gère ses politiques au sein d'une communauté d'une manière autonome au moyen d'un seuil maximal de tolérance du risque des requêtes d'acteurs externes. Ainsi, pour déterminer le seuil, une entreprise peut opter pour un mécanisme comme *Cost* (coût) [47], ou l'évaluation des dommages (*damages*) [154, 143]. Il existe même des travaux qui traitent les problèmes de définition de seuil dynamique et adaptatif comme [42, 41]. Dans le cadre de cette thèse, nous n'avons pas abordé ce challenge.

Conformément à nos objectifs, notre approche nous permet de prendre en compte des informations supplémentaires pour évaluer la probabilité de :

- l'usurpation d'identité, comme dans *l'exemple de motivation 2*, à travers la menace que reflète l'acteur en question (**cf.*(ex.2)**, *section* ***Exemple de motivation***, *chapitre* ***Problématique et motivations***).
- l'importance de la ressource, comme dans *l'exemple de motivation 3*, à travers l'*Impact*. Ce qui permet par ailleurs, de tenir compte de la sensibilité dynamique d'une ressource collaborative (**cf.*(ex.3)**, *section* ***Exemple de motivation***, *chapitre* ***Problématique et motivations***).
- les failles des infrastructures de sécurité, comme dans *l'exemple de motivation 4*, grâce à la *Vulnérabilité* (**cf.*(ex.4)**, *section* ***Exemple de motivation***, *chapitre* ***Problématique et motivations***).

6.4 Expérimentation

Nous avons effectué différentes expérimentations pour valider notre approche. En raison d'un manque de données de scénarios réels, nous avons d'abord simulé des requêtes d'accès, et nous avons calculé la valeur du risque de chacune de ces requêtes pour voir l'influence de notre approche sur un système de contrôle d'accès existant.

6.4.1 Implémentation

Tout d'abord, nous générons un ensemble de politiques de contrôle d'accès pour cinquante utilisateurs différents sur cinquante ressources différentes. Nous avons sélectionné les trois actions couramment utilisées, à savoir, *"Lire"*, *"Écrire"* et *"Exécuter"*. Les trois opérations *lire*, *écrire* et *exécuter* sont générées respectivement avec les proportions suivantes $\{0.7, 0.3, 0.4\}$. Notez que cette distribution est à l'image de plusieurs cas d'usage de notre partenaire *Brake France* [57] du projet *OpenPaaS* RSE.

Deuxièmement, nous définissons six méthodes d'authentification différentes. Ensuite, tel que défini dans *Définition* 2 nous attribuons un niveau de vulnérabilité à chacune d'elles. Nous illustrons dans le tableau 6.1 ces différentes méthodes et leurs valeurs de vulnérabilité respectives. Nous avons considéré que derrière chaque requête entrante, il y une identité d'un acteur, approuvée par l'un de ces mécanismes d'authentification. Nous convenons que l'attribution d'un niveau 0 de vulnérabilité à une authentification biométrique est controversée [58], mais nous avons fait cela uniquement dans le but d'illustrer le comportement de notre métrique d'évaluation du risque dans le cadre de nos expérimentations.

Enfin, nous générons aléatoirement 1500 requêtes de demande d'accès différentes. Chaque requête correspond à un tuple de 4 attributs, à savoir : { *utilisateur*, *ressource*, *action*, *méthode d'authentification* }. Essentiellement, nous nous sommes basés sur quatre scénarios : 10 % de rejets basés sur les politiques (fig. 6.2), 15 % (figure 6.3), 20 % (figure 6.4.) et 25 % (figure 6.5.). Cela nous permet de comparer le comportement de notre système de gestion du risque et son influence sur les

57. *Brake France* est une chaine de distribution de produits alimentaires qui a pour rôle dans le projet de tester la plateforme du réseau social.
58. http://www.net-security.org/secworld.php?id=8922

TABLE 6.1 – Méthodes d'authentification et leurs niveaux de vulnérabilité

Nom	**Description**	**Vulnérabilité**
None	Not authenticated user	1.0
PIN	Pin Code	0.8
L+P	Login and Password	0.6
OAuth	OAuth service	0.4
2F	Two factors	0.2
Bio	Biometric	0.0

décisions du mécanisme de contrôle d'accès dans différents scénarios. De plus, pour chacun de ces scénarios, nous avons calculé le risque avec deux pondérations, à savoir : avec une pondération de {1,1,1} (sur la gauche de chaque diagramme), et avec une pondération de {3,1,1} (sur la droite) pour respectivement l'*impact*, la *vulnérabilité* et la *menace*. La deuxième pondération permet d'atténuer l'effet de notre génération aléatoire, car elle met l'accent sur la valeur de l'impact (qui dépend des politiques).

Ainsi, pour chaque requête, nous avons une valeur de menace donnée, une valeur de vulnérabilité donnée et une valeur d'impact (calculée à base des politiques générées). Nous supposons que chaque demande est d'abord évaluée par la politique de contrôle d'accès pour voir si elle doit être acceptée ou pas. Ensuite, nous procédons à la deuxième phase de décision basée sur l'évaluation du risque de la requête en question, qui est comparé au seuil de tolérance de l'entreprise. Dans ce qui suit, nous décrivons les résultats obtenus.

6.4.2 Résultats

Nous voulons évaluer l'influence de notre système sur une politique de contrôle d'accès existante. Ainsi, l'idée est de voir le taux additionnel de requêtes qui seront rejetées par la valeur du risque comparée au seuil. Pour cela, nous avons fait des tests par rapport à différents seuils de risque. Par ailleurs, une autre information intéressante est la *cohérence* des risque-rejets. Plus précisément, nous allons observer le taux de requêtes qui seront rejetées par la politique et le seuil de risque.

Les résultats de notre simulation sont présentés sous forme de barres empilées, et ce, par rapport à différents seuils, allant de 0.4 à 0.9. Les figures doivent être lues de la sorte :

- **Gris clair** (la partie inférieure de chaque barre), la proportion des requêtes rejetées à base des politiques, *i.e.,* $\{Policy = Reject, Risk = Accept\}$.
- **Gris foncé** (la partie intermédiaire de chaque barre), la proportion des rejets cohérents entre les requêtes rejetées par la politique et celles par le seuil du risque, *i.e.,* $\{Policy = Reject, Risk = Reject\}$.
- **Gris normal** (la partie supérieure de chaque barre), la proportion des requêtes rejetées à cause du seuil du risque, *i.e.,* $\{Policy = Accept, Risk = Reject\}$.

Dans la figure 6.2 où le taux de rejet par les politiques est de 10 %, l'influence du seuil de risque

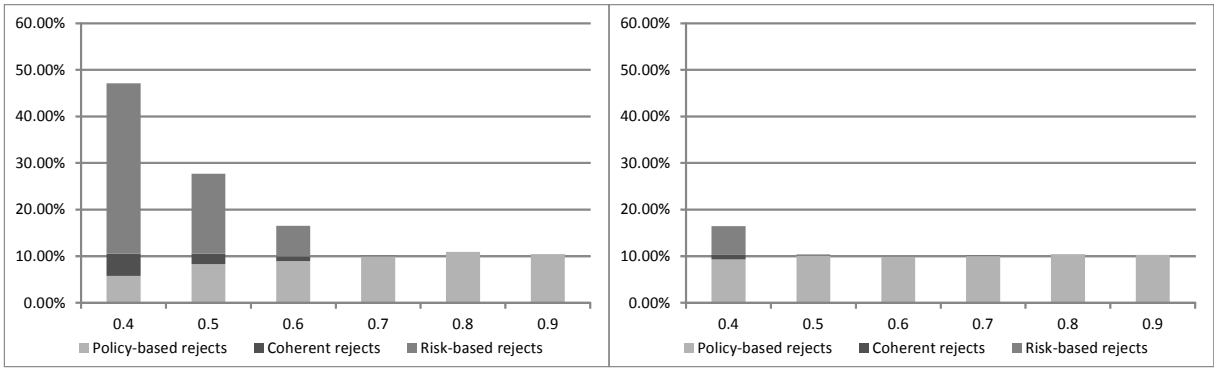

FIGURE 6.2 – Le ratio des requêtes rejetées par rapport à différents seuils de risque, pour une base de 10% de rejet par les politiques de contrôle d'accès.

est significative seulement pour un seuil considérablement bas, à savoir, 40% de rejets pour un seuil de 0.4. Cependant, avec un seuil plus élevé, le mécanisme d'évaluation du risque ne rejette plus beaucoup de requêtes. Le second scénario, avec une pondération focalisée sur l'impact, fait également ressortir cette observation. Nous constatons en outre que la pondération réduit significativement le taux de rejets par rapport au scénario non pondéré, seulement 7% pour le seuil de 0.4.

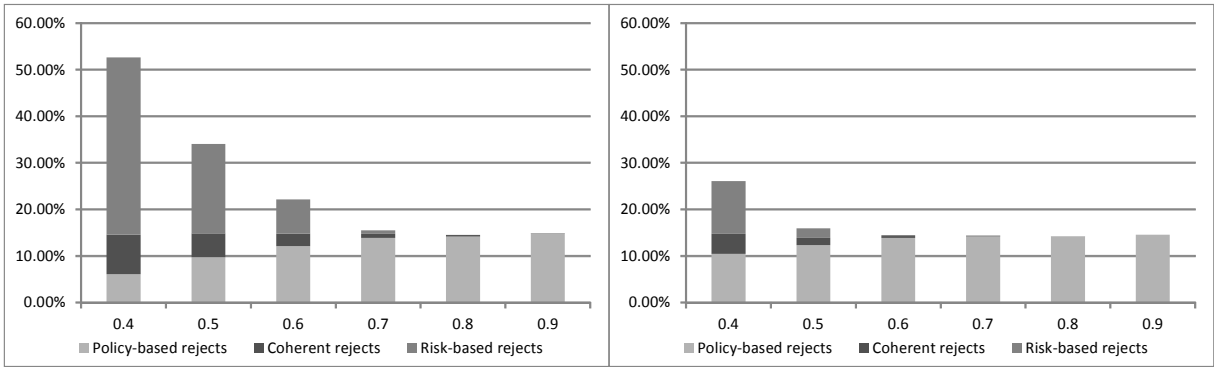

FIGURE 6.3 – Le ratio des requêtes rejetées par rapport à différents seuils de risque, pour une base de 15% de rejet par les politiques de contrôle d'accès.

Dans la figure 6.3, le taux de rejet par les politiques est de 15%. Comparé au scénario précédent, l'influence du risque est plus importante, à peu près 45% des requêtes rejetées par le seuil 0.4. De plus, plus le seuil du risque monte, plus de requêtes sont rejetées. Concernant la cohérence, elle est également plus importante que dans le scénario précédent. Nous remarquons par ailleurs que la deuxième pondération réduit le taux global de rejet de requêtes par le mécanisme du risque, en plus elle donne davantage de cohérence entre les taux de rejet politiques-risque que dans la figure Fig. 6.2.

Dans la figure 6.4 le taux de refus de requêtes s'élève à 20%. Dans cette configuration, l'influence du risque s'amplifie par rapport aux précédentes configurations (Fig. 6.2 et Fig. 6.3), environ 50% de

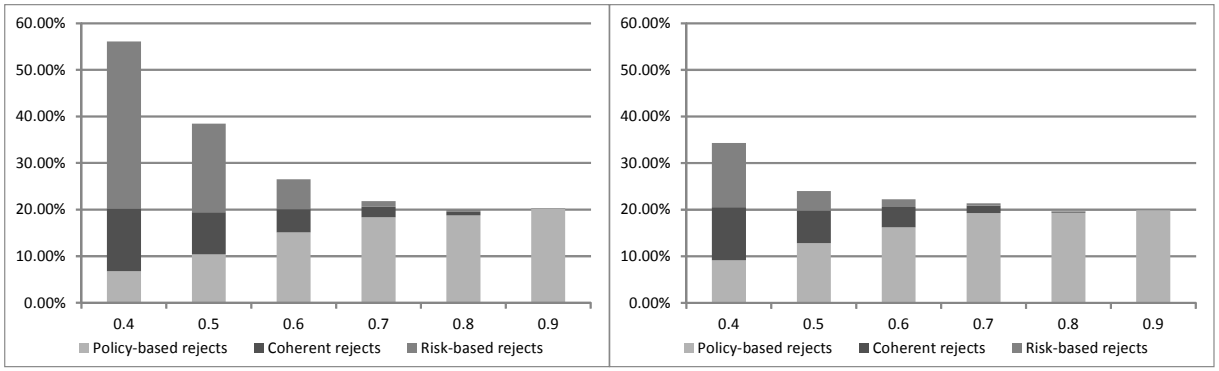

FIGURE 6.4 – Le ratio des requêtes rejetées par rapport à différents seuils de risque, pour une base de 20% de rejet par les politiques de contrôle d'accès.

requêtes non-validées par le seuil 0.4 de risque. De plus, nous remarquons que l'influence du risque s'étale jusqu'au seuil 0.7, et ce avec les deux pondérations. Par ailleurs, la cohérence est plus importante que précédemment. Cela est encore plus flagrant dans le scénario pondéré, où nous remarquons que les décisions basées sur les politiques et celles sur le risque sont davantage cohérentes.

Pour terminer, la figure 6.5 illustre le cas où 25% des requêtes sont rejetées par les politiques de contrôle d'accès. Cette figure montre en outre le taux de refus le plus élevé (environ 52%) parmi les 4 configurations. Nous constatons toujours que, d'une part, l'effet du risque continue à augmenter, et ce même pour des seuils élevés, et d'autre part, la cohérence est globalement en augmentation. Cependant, contrairement aux scénarios précédents, dans la figure 6.5 le mécanisme du risque continue à rejeter des requêtes même avec un seuil de 0.8. Par ailleurs, à l'image de la figure 6.4, avec la pondération de l'impact, la figure 6.5 montre également plus de cohérence entre les décisions politiques-risque.

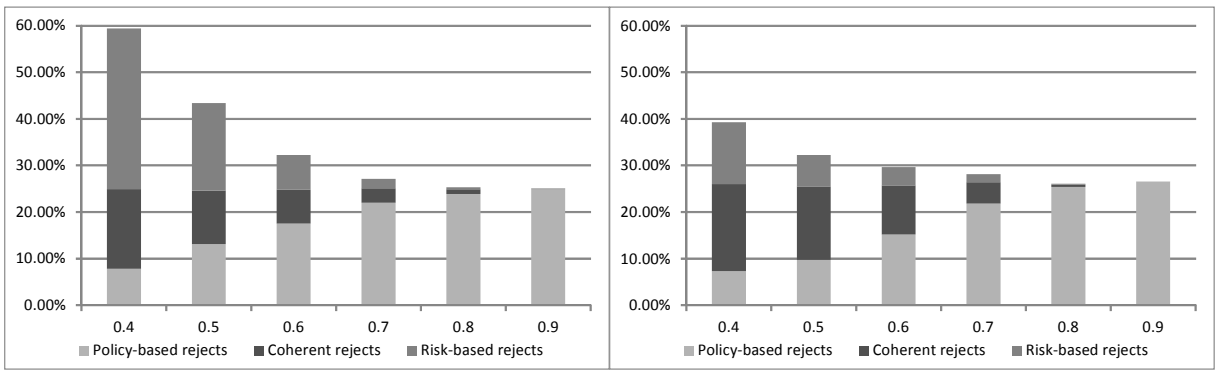

FIGURE 6.5 – Le ratio des requêtes rejetées par rapport à différents seuils de risque, pour une base de 25% de rejet par les politiques de contrôle d'accès.

6.4.3 Discussion

Notre analyse des résultats à travers les différentes figures précédentes nous a permis de tirer les conclusions suivantes :

- Moins le taux de rejets par les politiques est important, moins le risque sera influent. Ainsi, pour les systèmes où il est connu à l'avance que le taux de requêtes qui seront rejetées par les politiques de contrôle d'accès sera faible, le mécanisme de risque n'apportera pas une grande influence. En revanche, pour les environnements ouverts tels que les réseaux sociaux où le nombre de requêtes rejetées est souvent susceptible d'être considérablement élevé, l'influence du mécanisme d'évaluation de risque est bien plus présente, *i.e.*, globalement, il y a plus de requêtes rejetées à cause du risque qu'elles représentent, et ce même pour des seuils de risque élevés. Cela s'explique par la considération de l'importance (sensibilité) des ressources collaboratives. Car plus une ressource est confidentielle, moins d'acteurs seront autorisés à y accéder, et par conséquent plus de requêtes seront rejetées à son égard.
- L'observation précédente est accentuée avec la pondération qui donne plus d'importance à la valeur de l'impact des ressources par rapport aux reste des paramètres, à savoir, la menace et la vulnérabilité. Car nous avons remarqué que l'augmentation des rejets dus au risque est plus importante dans les systèmes où le taux de rejets par les politiques est plus élevé. Par ailleurs, la pondération améliore également la cohérence entre les décisions politiques-risque. Cela signifie que dans la formule pondérée, les décisions basées sur le risque sont conformes à celles prises à base des politiques.

Ces observations nous conduisent à conclure que notre système se comporte comme espéré. Le risque s'adapte bien au comportement du système basé sur politiques de contrôle d'accès définies par les acteurs de la communauté. En d'autres termes, il est plus prudent pour une entreprise d'être davantage sur ses gardes dans des environnements où il est assez fréquent que des acteurs externes tentent des accès illégaux aux ressources partagées.

De plus, l'influence globale de l'introduction de la métrique de risque semble correcte, tant que les seuils du risque restent raisonnables. Par cela, nous voulons dire que notre système ne va pas brusquement rejeter une grande quantité de requêtes. Cela semble cohérent avec le cas réel, où en général, une organisation fait confiance à ses utilisateurs pour l'établissement des politiques de partage de ressources collaboratives, et n'intervient seulement que dans les cas les plus critiques, *i.e.*, détection de menaces importantes.

Pour conclure, les expérimentations avec la formule pondérée de risque montre la cohérence de la métrique de risque que nous avons définie avec nos objectifs de sécurité. La cohérence des décisions à base du risque augmente par rapport aux décisions à base de politique d'une manière proportionnelle à la pondération qui a pour objectif l'atténuation des paramètres aléatoires des politiques produites pour nos expérimentations.

6.5 Conclusion

Dans ce chapitre, nous avons proposé une métrique de risque destinée à améliorer les performances, en matière d'efficacité de filtrage de requêtes malveillantes, des mécanismes classiques de contrôle d'accès.

Cette métrique d'évaluation du risque peut en effet être utilisée par dessus de n'importe quel autre mécanisme de contrôle d'accès. Notre approche est adapté aux environnements RSE, car elle permet aux entreprises de déléguer la définition des politiques de contrôle d'accès à ces utilisateurs (*user-centric approach*) tout en gardant le contrôle sur ces dernières d'une manière efficace, dynamique et abstraite.

En se basant sur les méthodologies standards de la gestion du risque, nous avons défini notre approche sur la base de trois paramètres qui tiennent compte des aspects suivants, l'impact de la ressource demandée, la menace de la requête reçue et la vulnérabilité de l'environnement collaboratif en question. Ajoutant à cela, un seuil et permet l'entreprise qui héberge la ressource demandée de rejeter certaines requêtes jugées, de sources malveillantes.

Nous avons défini de manière formelle les deux paramètres nécessaires à notre gestion du risque, à savoir l'impact et la vulnérabilité. Le premier reflète l'importance de la ressource à travers le nombre d'autorisations impliquant les acteurs et les actions. Le second paramètre est subjectif et permet de classifier les mécanismes d'authentification par ordre de fiabilité afin d'estimer les vulnérabilités de l'environnement collaboratif. Le dernier paramètre dans notre métrique d'évaluation du risque est la menace que reflète la requête reçue. L'évaluation de cette menace est basée sur la confiance du sujet de la requête. Par conséquent, la question qui reste à éclaircir concerne la manière dont sera évaluée cette confiance. C'est sur cette question que nous avons travaillé dans le cadre du chapitre suivant, à savoir, ***Confiance numérique***.

Chapitre 7

Confiance numérique

Sommaire

7.1 Introduction

Dans ce chapitre nous allons détailler notre approche dynamique d'évaluation de la réputation et la confiance numérique des sujets de collaboration au sein des communautés RSE. En premier lieu, nous allons introduire le contexte avec quelques définitions permettant d'élucider certaines notions et termes utilisés dans le cadre de ce chapitre. Ensuite, nous présenterons nos algorithmes d'évaluation de la réputation et de la confiance. Avant de conclure, nous illustrons l'efficacité de nos procédures d'évaluation de la réputation et de la confiance à travers une étude expérimentale dans laquelle nous observons l'évolution de la réputation et de la confiance par rapport aux comportements de sujets sur une succession de sessions collaboratives dans une communauté de collaboration.

7.2 Contexte

"La confiance, pense-t-on, ne va pas sans conditions. On ne peut l'accorder à n'importe qui les yeux fermés, ni réclamer d'autrui qu'il nous l'accorde aveuglément sans la dégrader en simple crédulité ... Il faut pour cela donner certains gages : le témoignage d'actes antérieurs, et une certaine qualité de l'attitude présente qui laisse à penser que les actes futurs seront de l'étoffe des précédents." C'est de ces

mots, utilisés par *Gildas Richard* pour donner une définition philosophique [163] de la confiance, que notre vision de la confiance s'inspire. En effet, cette définition met en évidence le lien indivisible et mutuel entre le comportement et la confiance, ainsi nous pourrons qualifier la confiance comme étant une *conséquence comportementale*. Par conséquent, trois mots clés permettent de cerner la question de confiance, à savoir le passé, le présent et le futur.

Dans le domaine informatique en général et celui de la sécurité collaborative en particulier, plusieurs chercheurs se sont posés la question sur la manière avec laquelle il est possible de modéliser la confiance numérique afin qu'elle soit avantageuse pour la qualité de la collaboration [33, 81, 12, 190, 104, 188]. Dans ce contexte, il est difficile de ne pas remarquer l'omniprésence du terme *"réputation"* qui peut être, d'une vision générale, considéré comme un synonyme de la confiance. Néanmoins, bien que étroitement liées et parfois confondues, les notions de réputation et de confiance manifestent certaines différences fondamentales, et ce notamment quand il s'agit de sécurité informatique dans les environnements collaboratifs.

- ***Réputation*** : une mesure réévaluée en continue au moyen d'un processus de supervision sur la base d'une ligne historique comportementale d'un sujet collaboratif.
- ***Confiance*** : notion plus abstraite qui permet de catégoriser (*i.e.* discrétionner) la valeur de la réputation. Peut être utilisée comme un indice de confiance sur lequel des évaluations ainsi que des éventualités peuvent être fondées entre sujet/système vis-à-vis d'un autre sujet.

7.3 Présentations conceptuelles de l'évaluation de la confiance et la réputation

L'évaluation de la confiance d'un sujet dans notre mécanisme est basée sur sa réputation. Cette dernière est mesurée à partir des données historiques collectées au cours de ses sessions collaboratives achevées. Une session est définie par un intervalle de temps durant lequel les interactions des utilisateurs auront lieu et leurs métadonnées, comme le nombre de tentative d'accès, les ressources utilisées, les permissions définies, *etc.*, sont enregistrées sous forme de fichiers *Logs*.

L'idée globale de notre mécanisme d'évaluation de confiance est la suivante. Pour chaque sujet, un indice de confiance est calculé à la fin de chaque session. Ainsi, nous suivrons l'évolution de cet indice de confiance par rapport aux sessions précédentes. Puis, sur la base de cette évolution, nous déterminons la manière dont sera évalué le comportement du sujet au cours de sa prochaine session. Nous considérons le comportement d'un sujet dans une communauté par rapport à ses demandes (requêtes) d'accès. Plus précisément, d'un point de vue de sécurité nous nous intéressons aux requêtes rejetées par le mécanisme de contrôle d'accès, que nous considérons comme des tentatives d'accès illégales aux ressources collaboratives.

Nous supposons que les requêtes de demande d'accès de chaque sujet, avec les décisions correspondantes sont collectées, agrégées et enregistrées dans un fichier *Log* du sujet, qui sera enregistré dans la base d'informations de la communauté en question.

D'un point de vue plus technique, notre mécanisme d'évaluation de la confiance est basé sur

la procédure suivante. Pour un sujet donné, *e.g. James*, le service *CTS* calcule à la fin de chaque session collaborative la réputation courante de *James* par rapport à son dernier *facteur de pénalité* et le *nombre de requêtes illégales* détectées. Ensuite le *CTS* mesure l'évolution du score de la *réputation courante* de *James* avec celui de sa *réputation passée* . Le score de réputation passée (ou initiale) est calculé sur la base de la moyenne des scores de réputations de toutes ses sessions achevées. Sur la base de cette évolution, le *CTS* met à jour (au niveau du *CIS*) le facteur de pénalité et ainsi l'indice de confiance du sujet *James*.

L'indice de confiance est en effet calculé sur la base d'un *facteur de pénalité*. Ce dernier s'applique sur le nombre de requêtes rejetées de *James* pour le pénaliser. Ainsi, la réputation courante est calculée à partir du dernier indice de confiance obtenu (*i.e.* mis à jour lors de la dernière session) et le nombre de tentatives d'accès non autorisées, collectées au courant de la session courante.

Par conséquent, le score de réputation est dynamique et change au cours des sessions. Ce changement est dû en premier lieu aux variations des changements comportementaux reflétées par le nombre de requêtes non autorisées établies par le sujet en question. En second lieu, cela est dû aux changements de l'indice de confiance qui peut changer en fonction de l'évolution (ou la dégradation) des scores de réputation du sujet en question.

7.3.1 Présentation formelle du mécanisme d'évaluation de confiance

Dans notre contexte, nous considérons que la réputation d'un sujet diminue et/ou augmente à un taux proportionnel au nombre de ses actions illégales, à savoir le nombre de demandes refusées. Par conséquent, une interception rapide d'un comportement suspect d'un sujet est nécessaire pour qu'elle soit prise en compte dans le processus de contrôle d'accès. C'est pourquoi, nous formalisons notre système d'évaluation de la réputation $\boldsymbol{tr}$ avec une fonction *exponentielle* [190] qui prend comme paramètres par rapport à un sujet $\boldsymbol{S}$:

- le nombre de demandes refusées $\boldsymbol{nbDeny}$ collectées à la fin de la session courante,
- le dernier facteur de pénalité $\boldsymbol{\rho}$.

L'algorithme 1 décrit notre fonction d'évaluation de la réputation d'un sujet.

Algorithm 1 Évaluation de la réputation

```
1: function ReputationComputing(S, ρ, nbDeny)
2:     tr = exp(−ρ ∗ nbDeny);
3:     retourner tr ;
4: fin function
```

La confiance (*i.e.* indice de confiance) peut être calculée sur la base du dernier facteur de pénalité obtenu. En effet, la confiance est l'inverse du facteur de pénalité. L'algorithme 2 illustre la fonction de calcul de la confiance $TrustLevel$.

Algorithm 2 Évaluation de la confiance

1: **function TrustComputing**(S, ρ)
2: $TrustLevel = 1 - \rho$;
3: **retourner** $TrustLevel$;
4: **fin function**

Dans le cadre de notre évaluation dynamique du comportement d'un sujet de collaboration, l'indice de confiance est en effet un ensemble d'étiquettes permettant de décrire le niveau de confiance qu'accorde le système à un sujet [12]. En effet, il s'agit d'un ensemble de valeurs discrètes permettant de spécifier les catégories de confiance qui peuvent être attribuées aux sujets de la communauté. Par exemple, *très fiable, fiable, normal, non-fiable, suspicieux*. À chaque étiquette de l'ensemble correspond une valeur réelle.

La valeur du facteur de pénalité $\boldsymbol{\rho}$ peut changer à la fin de chaque session. Ces variations (augmentation/diminution) de $\boldsymbol{\rho}$ obligeront un sujet à prêter attention à son comportement. En effet, le comportement d'un sujet a un impact direct sur l'évolution de $\boldsymbol{\rho}$. Tant que, le sujet ne cherche pas à accéder à des ressources non autorisées, le $\boldsymbol{\rho}$ appliqué restera faible.

Pour calculer le $\boldsymbol{\rho}$ qui sera appliqué pour la prochaine session d'un sujet, nous devons d'abord calculer le score initial de sa réputation $\boldsymbol{tr_0}$, et ce sur la base des scores obtenus dans ses sessions passées (*i.e.* historiques de collaboration) stockés au niveau du *CIS*. Ensuite, nous avons besoin de mesurer le taux d'évolution $\boldsymbol{\lambda}$ entre le score de réputation initial et le score de réputation $\boldsymbol{tr}$ courant, *i.e.* calculé à la fin de la dernière session achevée. Plus précisément, supposons que $\boldsymbol{n}$ est le nombre de sessions de collaboration du sujet. Pour calculer $\boldsymbol{tr_0}$, le *CTS* calcule la moyenne des scores des réputations obtenus au courant des $\boldsymbol{n-1}$ sessions consommées par le sujet. Cependant, dans le calcul de moyenne, le *CTS* donne plus d'importance (pondération par 2) à la dernière expérience (*i.e.* session $\boldsymbol{n-1}$) par rapport à celles qui restent ($\boldsymbol{1..n-2}$).

Ensuite, pour calculer la taux d'évolution $\boldsymbol{\lambda}$ de la réputation, le *CTS* mesure le taux de variation entre la réputation initiale $\boldsymbol{tr_0}$ et la réputation courante $\boldsymbol{tr}$ en utilisant une fonction exponentielle décroissance/croissance. Ainsi, si $\boldsymbol{tr}$ est plus élevé que $\boldsymbol{tr_0}$, alors $\boldsymbol{\lambda}$ sera une valeur positive, sinon $\boldsymbol{\lambda}$ sera une valeur négative. Par conséquent, si l'évolution est positive ($\boldsymbol{\lambda} > 0$), elle va réduire le facteur de pénalité $\boldsymbol{\rho}$. Inversement, si l'évolution est négative ($\boldsymbol{\lambda} < 0$), $\boldsymbol{\rho}$ sera aggravé (*i.e.* augmentation du facteur de pénalité).

Dans le but de se rapprocher de cas réels, il est logique que les variations dans la sévérité de pénalisation ne soit pas la même pour un sujet loyal et un sujet malhonnête. Plus précisément, l'indice de confiance d'un sujet malhonnête ne doit pas évoluer (baissant la sévérité de pénalisation) dans la même proportion qu'un sujet loyal. Inversement, un sujet loyal doit être sanctionné avec plus de sévérité qu'un sujet déjà suspect. En effet, un sujet loyal qui change brusquement son comportement peut être un pirate, par conséquent, la sanction doit être sévère afin de réagir rapidement et protéger les ressources collaboratives.

Cela signifie que la catégorie de la confiance du sujet [12], interprétée par sa dernière valeur du facteur de pénalité, est directement impliquée dans l'évolution de $\boldsymbol{\rho}$. Par conséquent, nous pondé-

rons le $\boldsymbol{\lambda}$ obtenu avec le niveau de confiance du sujet, (*i.e.* $\mathbf{1-\rho}$). En outre, nous avons aussi besoin d'adoucir l'éventuel changement brusque et élevé de $\boldsymbol{\lambda}$, donc nous divisons le résultat obtenu par le *facteur de sévérité de la communauté* $\boldsymbol{\varsigma}$. Ce dernier est un paramètre subjectif défini par l'administrateur de la communauté. Plus ce paramètre est élevé, plus il sera difficile pour les sujets de diminuer leurs $\boldsymbol{\rho}$ respectifs.

Algorithm 3 Taux d'évolution du facteur de pénalité

1: **function TrEvolFact**(S, tr_0, tr, ς, ρ)
2: $\lambda = [(\ln(tr/tr_0)/2) * (1-\rho)]/\varsigma$;
3: **retourner** λ ;
4: **fin function**

Le calcul du nouveau $\boldsymbol{\rho}$ est basé sur le résultat du dernier taux d'évolution $\boldsymbol{\lambda}$. Pour cela, à l'image de la réputation et la confiance, nous avons deux sortes de facteurs de pénalité, un *discret* $\boldsymbol{\rho}$, et l'autre *continu* $\boldsymbol{\varrho}$. Ce dernier ($\boldsymbol{\varrho}$) est utilisé pour analyser les changements en continu du comportement d'un sujet. Tandis que $\boldsymbol{\rho}$ est directement appliqué pour sanctionner les éventuelles actions illégales du sujet. Ainsi, nous calculons le facteur de pénalité $\boldsymbol{\rho}$ comme suit. Tout d'abord, nous soustrayons le taux d'évolution courant $\boldsymbol{\lambda}$ du dernier facteur de pénalité continue $\boldsymbol{\varrho}$. Ensuite nous procédons à la discrétisation du $\boldsymbol{\varrho}$ obtenu par rapport à l'intervalle discret $\boldsymbol{DiscP[\,]}$. En outre, après un comportement suspicieux, plus les valeurs discrètes sont proches, moins la catégorie de confiance descend.

Algorithm 4 Calcul du facteur de pénalité

1: **function penaltyFactor**(S, ϱ, ρ, λ)
2: $\varrho = \varrho - \lambda$;
3: $\rho =$ **PenaltyDiscritization**($\varrho, DiscP[\,]$) ;
4: **retourner** ρ ;
5: **fin function**

Avec notre fonction qui calcule $\boldsymbol{\lambda}$, les valeurs discrètes de l'ensemble $\boldsymbol{DiscP[\,]}$ doivent toujours être $\in \mathbf{]0,1[}$, parce que si $\boldsymbol{\rho}$ est égal à $\mathbf{0}$, alors aucune pénalité ne sera appliquée dans la communauté. Toutefois, si $\boldsymbol{\rho}$ est égal à $\mathbf{1}$, le taux d'évolution $\boldsymbol{\lambda}$ converge vers la valeur $\mathbf{0}$. C'est pourquoi, il est recommandé d'utiliser l'intervalle réel $\boldsymbol{DiscP[\,]} \subseteq \mathbf{]0,1[}$, sauf dans le but de désactiver le mécanisme d'évaluation de confiance.

La procédure de discrétisation est utilisée pour empêcher un sujet de changer immédiatement son facteur de pénalité après une session de collaboration. Nous utilisons pour cette procédure un algorithme de recherche dichotomique pour trouver la valeur discrète la plus proche du $\boldsymbol{\rho}$ dans l'intervalle discret. Quand la valeur $\boldsymbol{\varrho}$ dépasse la borne supérieure ou inférieure de l'ensemble discret, la valeur de $\boldsymbol{\rho}$ est réduite aux valeurs inférieures et/ou supérieures de $\boldsymbol{DiscP[\,]}$ respectivement. La discrétisation permet par ailleurs d'aligner tous les sujets de la communauté sur les mêmes facteurs de pénalisation.

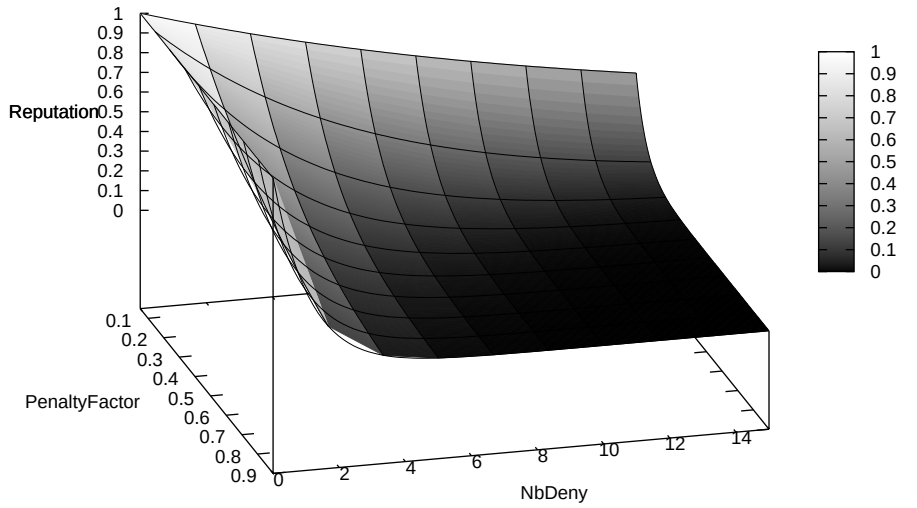

FIGURE 7.1 – Fonction d'évaluation de la confiance.

7.3.2 Illustration

À titre d'exemple, nous utilisons l'ensemble discret suivant, $\boldsymbol{DiscP[\,] = \{0.05, 0.1, 0.5, 0.9\}}$. Cet ensemble peut être interprété avec les étiquettes suivantes, *très fiable, fiable, non-fiable, suspicieux*. En outre, nous supposons que le nombre maximum de tentatives d'accès refusées est de $\boldsymbol{maxCommNbDeny = 15}$. Le résultat de cette configuration de notre approche d'évaluation de confiance est illustré dans la figure 7.1. Comme nous voyons dans le graphique, plus le facteur de pénalité $\boldsymbol{\rho}$ augmente, plus la valeur de confiance baisse par rapport au nombre des requêtes non autorisées.

7.4 Étude expérimentale

Nous avons intégré notre mécanisme d'évaluation de confiance dans la plateforme *OpenPaaS*. Cependant, nous manquons actuellement de réelles traces d'expériences utilisateurs. Par conséquent, pour évaluer notre approche de l'évaluation de la confiance, nous avons fait quelques tests basés sur deux expériences de deux sujets, *Alice* et *Oscar*. Nous avons ensuite simulé le comportement de ces deux sujets de telle sorte qu'*Alice* ait un comportement acceptable et stable, et que le comportement d'*Oscar* soit instable et parfois suspect.

La durée de session pour la collecte des traces des sujets d'une communauté est un paramètre subjectif d'administration [59], *e.g.* chaque heure, jour, mois, trimestre ou plus. De plus, nous supposons que le nombre de requêtes rejetées qui sont collectées à la fin de chaque session est significatif et non corrompu. Par exemple si la durée de session est délimitée par heure, et qu'un sujet ne fait aucune

59. Peut être un paramètre de création de communauté.

interaction pendant une ou plusieurs sessions, ses traces (très positives) pour cette session ne seront pas prises en considération pour l'évaluation de l'évolution de sa réputation. Dans le cadre de cette thèse, nous n'adressons pas cette problématique, car cela peut dépendre de plusieurs paramètres, comme le temps passé par session, la fréquence d'interaction, *etc*. Ce point sera mieux traité une fois que l'on dispose de vraies statistiques d'expérience réelles d'utilisateur sur la plate-forme *OpenPaaS*. Il est de même pour les valeurs initiales des réputations ainsi que le niveau de confiance. En effet, nous nous basons sur des valeurs moyennes, *i.e.* pas très pénalisantes ni favorisantes.

Pour initialiser les expériences de nos deux sujets *Alice* et *Oscar*, nous considérons que leurs valeurs initiales de confiance sont $\boldsymbol{tr_0 = 0.5,\ tr = 0.6,\ \rho = 0.1, \varrho = 0.1}$. Afin de simplifier la compréhension de notre étude expérimentale, nous avons fixé le facteur de sévérité de la communauté à $\boldsymbol{\varsigma = 1}$, ce qui signifie que l'évolution de l'indice de confiance $\boldsymbol{\rho}$ sera très rapide.

Dans les deux figures 7.3 et 7.2 le graphe à ligne discontinue montre l'évolution du facteur de pénalité $\boldsymbol{\rho}$, tandis que celui à ligne continue indique l'évolution de la réputation du sujet.

Comme *Alice* est supposée être une personne honnête, nous avons généré son comportement au moyen d'une fonction aléatoire de **3** à **6** requêtes rejetées. Quant à *Oscar*, son comportement est instable. Pour cela, nous avons défini plusieurs vecteurs comportementaux pour *Oscar*.

Dans le premier vecteur, *Oscar* a un comportement normal pour un certain nombre de sessions successives (**Normal**). Pour la deuxième période, *Oscar* améliore son comportement redevient exemplaire, *i.e.* nombre négligeable de requêtes rejetées (**Normal2**). Cela devrait réduire le facteur de pénalité d'*Oscar*. Après, un comportement suspicieux d'*Oscar* est détecté sur une suite de sessions par une fréquence élevée de requêtes non-autorisées sur les ressources partagées (**Mauvais**). Enfin, le comportement d'*Oscar* revient à l'état normal.

Normal	5	10	9	5	8	9	7	8	5	6	8	
Normal2	4	3	2	5	4	1	4	2	3	5	1	2
Bon	4	3	5	2	2	1	1	0	2	1	1	
Mauvais	10	4	3	2	14	4	1	2	11	3	1	2

TABLE 7.1 – Le comportement d'*Oscar*

7.4.1 Discussion

La figure 7.3 illustre les changements comportementaux d'*Oscar* à travers un certain nombre de sessions collaboratives. Le fragment à ligne double désigne la bonne période pour *Oscar*, et l'autre fragment à ligne simple désigne la mauvaise période.

Ainsi, notre analyse du graphique est la suivante. Avant la bonne période où *Oscar* se comporte d'une manière normale (**Normal**), sa réputation est plus au moins stable. Ensuite, durant la bonne période où le comportement d'*Oscar* est exemplaire, nous remarquons que son facteur de pénalité baisse pénalisant avec moins de sévérité les mauvaises actions relevées, et par conséquent la réputation d'*Oscar* est améliorée. Plus tard, quand le système de supervision détecte qu'*Oscar* entreprend soudainement un comportement suspicieux en essayant d'accéder avec une fréquence élevée à des

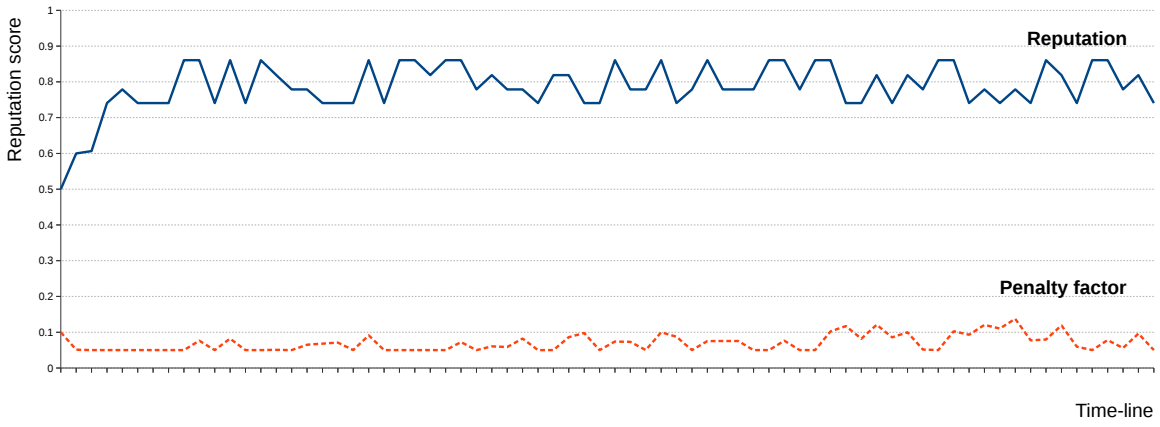

FIGURE 7.2 – Évolution de la réputation d'un sujet à comportement stable et honnête.

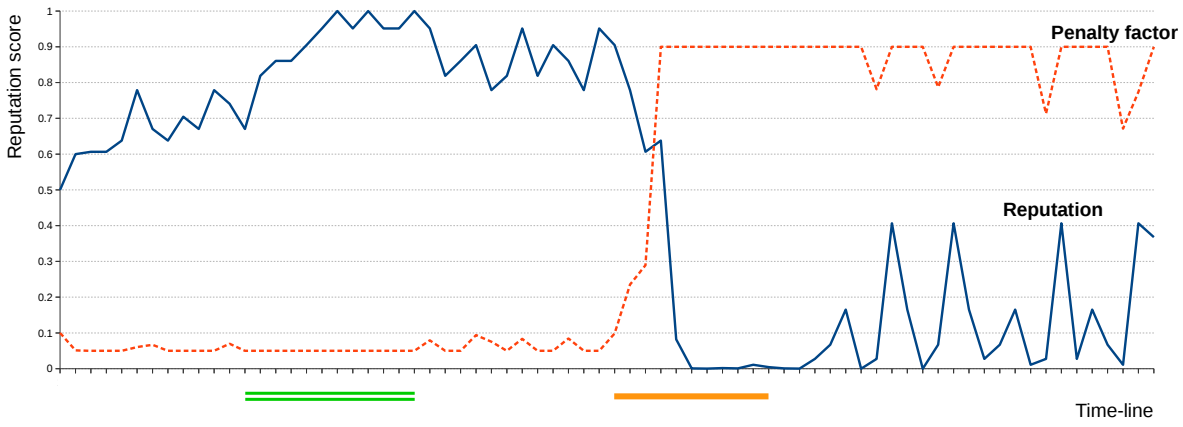

FIGURE 7.3 – Évolution de la réputation d'un sujet à comportement instable et suspicieux.

ressources qui lui sont non autorisées, le facteur de pénalité augmente considérablement provoquant ainsi une importante baisse de la réputation d'*Oscar*. Après la mauvaise période, *Oscar* tente de se racheter, et reprend un comportement très honnête (**normal2**). Ce dernier comportement est censé améliorer la réputation d'*Oscar*. Cela dit, la réputation d'*Oscar* n'est pas la même que dans la bonne période (ligne double) d'avant la détection du comportement malicieux.

De l'autre part, la figure 7.2 montre les variations de la réputation d'*Alice*. Comme nous le voyons, le comportement d'*Alice* est stable, cela se reflète par conséquent dans ses valeurs de réputation à travers ses sessions collaboratives.

7.5 Conclusion

Dans ce chapitre, nous avons détaillé notre approche dynamique d'évaluation de la réputation et de la confiance vis-à-vis des sujets de collaboration au sein des communautés *OpenPaaS*. Notre évaluation de la confiance est basée sur la supervision de l'évolution de la réputation d'un sujet au fil des sessions collaboratives. Quant à la réputation, elle se focalise sur le comportement du sujet en question en pénalisant ses actions collaboratives négatives, à savoir les requêtes illégales de demande d'accès aux ressources partagées de la communauté. La pénalisation d'un sujet est fondée sur son indice de confiance mis à jour à la fin de chaque session collaborative, et ce en fonction de l'évolution de la réputation courante du sujet par rapport à ses scores de réputation antérieurs.

Cette approche contraint les sujets à faire attention à leurs comportements respectifs au sein des communautés de collaboration. Elle permet en outre d'anticiper une usurpation d'identité à travers tout changement comportemental brusque et bloquer ainsi l'accès au sujet en question. Le blocage d'accès se fait au moyen du mécanisme de gestion du risque (présenté dans le chapitre (**cf. Gestion du risque**) qui inclut cette métrique d'évaluation de confiance comme un paramètre d'estimation de la menace d'une requête reçue. Le mécanisme de gestion du risque est le moteur principal des politiques entreprises destinées à superviser les politiques de partage de ressources par les sujets de collaboration. Par conséquent, la confiance devient un paramètre clé dans notre mécanisme de contrôle d'accès *OpenPaaS*.

Notre proposition d'un mécanisme de sécurité des ressources collaboratives inclut tous les composants précédemment détaillés, à savoir le mécanisme d'authentification interopérable, les politiques de partage de ressources, les politiques entreprises qui incluent la gestion du risque et de la confiance. Ces composants ont été validés et mis en œuvre dans le cadre du projet *OpenPaaS* RSE. Dans le chapitre suivant, nous allons aborder l'aspect technique de notre conception en détaillant la mise en œuvre différents composants logiciels que nous avons intégrés dans *OpenPaaS* RSE, à savoir l'*authentification*, l'*autorisation* et l'*audit*.

Chapitre 8

Implantation des composantes de sécurité

Sommaire

Dans ce chapitre nous allons présenter les différentes APIs que nous avons développées et intégrées dans le cadre de la plate-forme *OpenPaaS* RSE. Il s'agit de trois composants proposés sous forme de service web pour la gestion de l'authentification, de l'autorisation et de l'audit (supervision). Bien évidemment, l'implantation de ces trois services est intégralement basée sur les concepts de sécurité présentés dans les chapitres précédents.

Le chapitre est ainsi organisé. D'abord, nous présentons une vue conceptuelle globale de notre système au moyen d'un diagramme de classe UML. Ensuite, nous rappelons brièvement l'idée fondatrice de conception de chaque composant, suivi de son implantation avec des détails sur l'aspect technologique de mise en œuvre. Enfin nous présenterons les performances pour montrer, outre l'efficacité, l'optimalité de notre API de sécurité.

8.1 OpenPaaS RSE vue abstraite

L'architecture de la plateforme OpenPaaS vue sous l'angle de sécurité est illustrée dans la figure 8.1. Les entités principales sont le sujet (*Member*), la communauté (*Community*) et les ressources

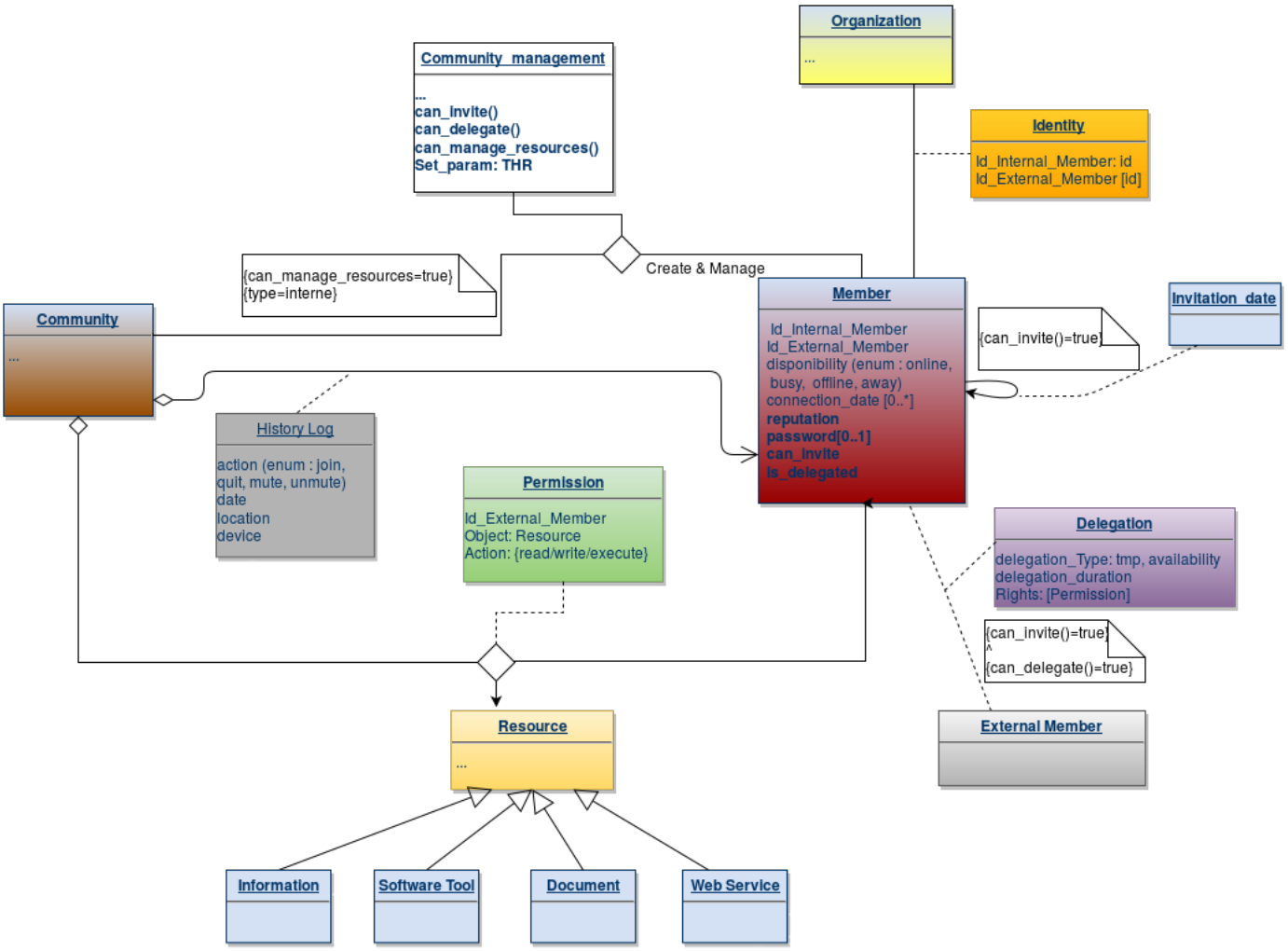

FIGURE 8.1 – Architecture globale de plateforme de sécurité d'OpenPaaS

(*informations, outils logiciel, document, service web*). Nous avons également l'entreprise (*Organization*) à laquelle appartient un sujet et dans laquelle est définie son identité. En outre, nous avons les permissions sur les ressources partagées par un sujet dans une communauté ainsi que les délégations entre sujets. Enfin, nous avons la classe *History Log* dans laquelle est enregistré l'historique d'interactions (les actions, les dates *etc*) d'un sujet.

8.2 Gestion des identités numérique –Authentification–

Notre principal objectif dans la gestion des identités numérique est de permettre à chaque entreprise de préserver son mécanisme d'authentification préféré et de pouvoir, d'un côté vérifier l'authenticité d'identité d'utilisateurs externes, et d'un autre côté permettre à ses utilisateurs internes de pouvoir s'authentifier auprès d'autres entreprises n'utilisant pas forcément le même mécanisme d'authentification. Pour ce faire, nous nous sommes basés sur l'outil *LemonLDAP-NG* (**cf. section 4.2.3.2,**

Chapitre Architecture et gestion des identités numériques).

8.2.1 Résumé de solution proposée

LemonLDAP-NG est installé en tant que service web au niveau de chaque entreprise et chaque communauté faisant partie du RSE. En effet, une forme de fédération qui consiste à déléguer la vérification de l'identité d'un acteur au gestionnaire d'authentification de la communauté est mise en place entre ce dernier et ceux des entreprises.

Le processus d'authentification se base sur les modules *LLdap-NG* suivants. Le *Portail LLdap* fait office de client SSO. Le module *AuthProxy* sert à transférer d'un *Portail* d'authentification à un autre les jetons/certificats d'identité pour vérification vis-à-vis de la base *LDAP* distante.

Grâce à ces deux composants logiciels, le gestionnaire d'authentification d'une communauté peut vérifier l'authenticité du certificat/jeton d'identité d'un acteur donné après la réception d'une réponse sous forme de *cookie* de la part du fournisseur d'identités de l'entreprise du sujet.

Techniquement, nous nous sommes basés en premier lieu sur une architecture Credential-based [32]. Dans ce type d'architecture, les Credentials sont utilisés par un acteur afin de prouver son identité auprès d'un fournisseur de services. Cependant, nous avons étendu l'architecture Credentials-based avec une interaction entre le fournisseur d'identités (entreprise) et le fournisseur de services (communauté) afin de vérifier que l'identité de l'acteur qui présente les *Credentials* n'a pas été usurpée. La procédure de vérification des Credentials se fait du côté du fournisseur d'identités une fois que ces derniers lui sont transférés par un fournisseur de services. Cette procédure de vérification de *Credentials* est basée sur deux types d'identités, à savoir privée (interne) et publique (externe). Une fois les *Credentials* vérifiés et validés, le fournisseur d'identités de l'utilisateur répond au fournisseur de services par un jeton sous forme de cookie. Dans le cas où l'authentification échoue (utilisateur non authentifié), le cookie sera vide.

8.2.2 Implantation

Pour tester le prototype nous avons opté pour le mécanisme d'authentification Login/Password avec des comptes utilisateurs prédéfinis dans *LemonLDAP-NG*.

- Nous avons commencé par installer touts les pré-requis logiciels nécessaires au bon fonctionnement de LemonLDAP-NG :

apt-get install apache2 libapache2-mod-perl2 libapache-session-perl libnet-ldap-perl libcache-cache-perl libdbi-perl perl-modules libwww-perl libcache-cache-perl libxml-simple-perl libsoap-lite-perl libhtml-template-perl libregexp-assemble-perl libjs-jquery libxml-libxml-perl libcrypt-rijndael-perl libio-string-perl libxml-libxslt-perl libconfig-inifiles-perl libjson-perl libstring-random-perl libemail-date-format-perl libmime-lite-perl libcrypt-openssl-rsa-perl libdigest-hmac-perl libclone-perl libauthen-sasl-perl libnet-cidr-lite-perl libcrypt-openssl-x509-perl libauthcas-perl libtest-pod-perl libtest-mockobject-perl libauthen-captcha-perl libnet-openid-consumer-perl libnet-openid-server-perl libunicode-string-perl libconvert-pem-perl libmouse-perl,

- ensuite, nous avons installé LemonLDAP-NG
  `http://lemonldap-ng.org/documentation/quickstart`
- enfin, nous avons configuré l'accès SOAP à LemonLDAP-NG
  `http://lemonldap-ng.org/documentation/latest/soapsessionbackend`.

Le nom de la méthode qui nous permet de récupérer le Cookie est *getCookies* se trouve dans la classe *AuthenticationPortTypeProxy* dans le package *Lemonldap*. Nous avons obtenu ce package grâce au fichier de description de service *portal.wsdl* fourni par *LemonLDAP-NG*. La méthode *getCookie* prend en paramètres un nom d'utilisateur et un mot de passe, et si ces paramètres sont valides elle retourne un Cookie non null, sinon la valeur du Cookie sera null.

Le client *REST* qui se charge d'interroger LemonLDAP-NG et récupérer le Cookie est illustré dans la figure 8.3. Ce client établit un appel REST via une requête HTTP sur le service d'authentification *AuthenticationService*. Au niveau du service d'authentification *AuthenticationService* qui est basé sur le package LemonLDAP-NG nous avons défini la méthode *authentication* illustrée dans la figure8.2. Cette méthode permet d'établir une connexion avec LemonLDAP-NG et récupérer la valeur du Cookie [60]. Cette méthode nous l'avons exposé pour qu'elle soit accessible par un appel en GET (REST) depuis le service d'application ou celui du contrôle d'accès.

Dépendances

La liste des dépendances nécessaires pour l'utilisation de LemonLDAP-NG sous *maven* est la suivante :

— *apache-jakarta-commons-discovery*
— axis-client
— commons-logging-1.2
— mail-1.4.7
— wsdl4j-1.5.2
— javax.mail-1.5.2

60. La vérification d'authentification de l'utilisateur se fait au niveau du service *AuthenticationService* grâce à la variable booléenne *isAuthenticatedUsed*, par conséquent, au niveau du client on pourrait récupérer directement la valeur de *isAuthenticatedUsed* pour vérifié si l'utilisateur est bien authentifié (ou pas) dans le cas où on voudrait pas faire un test supplémentaire

```java
package boundary;

import Lemonldap.AuthenticationPortTypeProxy;
import java.rmi.RemoteException;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.Response;

/**
 *
 * @author ahmed
 */
@Path("authentication")

public class Authentication {

    @GET
    public String authentication(@QueryParam("subject") String subject,
                                 @QueryParam("password") String password)
                                                        throws RemoteException{

            boolean isAuthenticatedUsed = false;
            String authenticationCookie = " ";
            AuthenticationPortTypeProxy a = new AuthenticationPortTypeProxy();
            authenticationCookie = a.getCookies(subject, password).getCookies().getLemonldap();

            if (authenticationCookie != null) {
                isAuthenticatedUsed = true;
            }
        return authenticationCookie;
    }
}
```

FIGURE 8.2 – Méthode de l'authentification

```java
    public String authentication(String subj, String psw) throws RemoteException {
        Client client = ClientBuilder.newClient();
        WebTarget target = client.target("http://localhost:8080/AUTHENTICATIONService/
                                         resources/authentication?subject="+subj+"&password="
                                                                          +psw);
        this.RESTCall = target.request().get(String.class);
        return RESTCall;
    }
```

FIGURE 8.3 – Client authentification

— activation-1.1

— javaee-api-7.0

Afin de faciliter l'installation et le test de LemonLDAP-NG, des comptes de démonstration on été mis en place. Pour la gestion des comptes utilisateurs, une documentation plus détaillée expliquant la procédure à suivre sur

http://lemonldap-ng.org/documentation/latest/authdemo.

8.3 Contrôle d'accès dans OpenPaaS –Autorisation–

Une fois le sujet authentifié, sa requête sera transférée et évaluée par le service d'autorisation. Le modèle d'autorisation que nous avons proposé est fondé sur le modèle ABAC (Attribute Based Access Control) avec une implantation formelle basée sur la logique temporelle *Event-Calculus*. Nous utilisons le raisonneur logique DEC-Reasoner [61] pour l'évaluation des patrons de contrôle d'accès que nous générons automatiquement à l'aide d'un pilote *Java* (exposé en tant que service Web). La figure 8.4 donne un aperçu sur le scénario de déroulement du processus de contrôle d'accès que nous avons conçu pour *OpenPaaS*. Les différentes étapes du shémas seront expliquées dans la sous-section suivante.

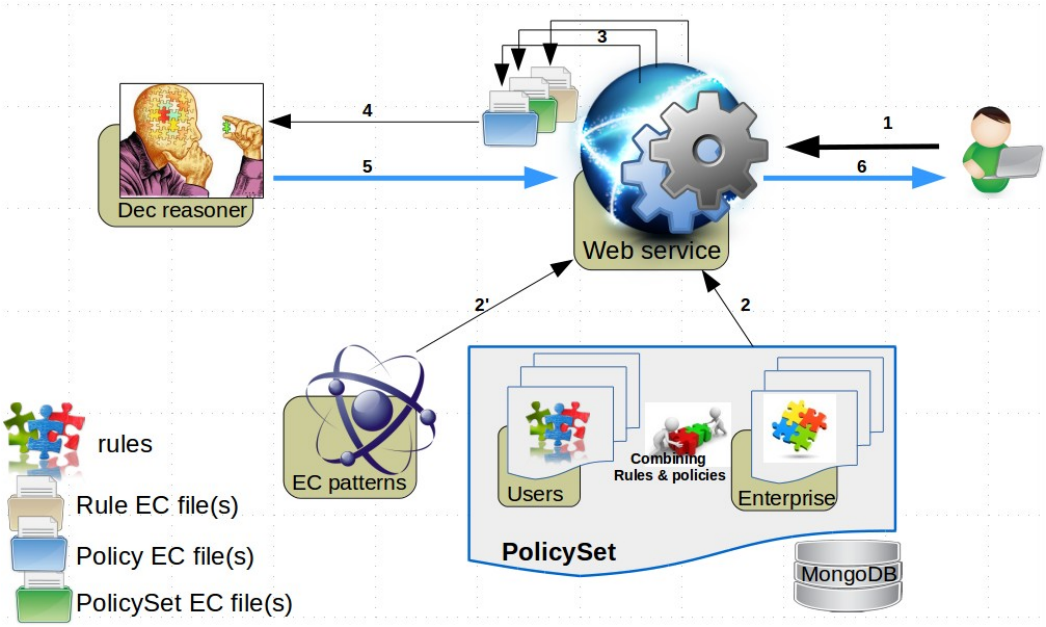

FIGURE 8.4 – Architecture globale de service de contrôle d'accès d'OpenPaaS

8.3.1 Résumé de la solution proposée

Comme le montre l'architecture dans la figure 8.4, le principal composant est le service web *Community-Decision-Service* (CDS) qui interagit avec le conteneur des règle, politiques et policySets le *Community-Administration-Store* (CAdS). Les principales étapes du processus sont :

- Définir des règles, politiques et policiesSets, ensuite les insérer dans la base de données *MongoDB*,
- récupérer la requête, les règles, les politiques et le policiesSet depuis la base de données *MongoDB*, ensuite les utiliser pour générer les fichier event-calculus *"file.e"* en se basant sur les patrons *Event-Calculus*,

61. http ://decreasoner.sourceforge.net/

- transférer les fichiers *"file.e"* au raisonneur logique *Dec-Reasoner* pour récupérer le résultat final après le processus du raisonnement.

Pour chaque étape du processus nous avons **respectivement** défini les classes : *AuthzStore, SaaSPatterns* et *SaaSResource*.

8.3.2 Implantation

Dans cette partie nous allons détailler la conception de chacune des classes *AuthzStore, SaaSPatterns* et *SaaSResource*.

8.3.2.1 Authorization store

@Path("authzStre") : class AuthzStore

Cette partie consiste en la gestion des composants de contrôle d'accès, à savoir les règles, politiques et les ensembles de politique (PolicySet). La gestion se fait par la mise à jour (*i.e* insertion, modification et suppression) de la base de données *MongoDB* en question. Nous avons exposé toutes les méthodes de gestion de la base donnée en *REST* pour que la mise à jour de la base soit plus simple via des requêtes *JSON*[62] depuis un client *REST*.

Insertion

Le modèle JSON pour l'insertion de **règle** : figure 8.5
→ *@POST, @Path("PostPath"), public Response insertRule(String msg)*

```
{
    "Subject" : userId,
    "Object" : resourceId,
    "Action" : { //the 4 actions have to be defined -> no default value
       "GET" : true,
       "POST" : true,
       "PUT" : false,
       "DELETE" : false
    },
    "RuleEffect": "Permit" || "Deny"
}
```

FIGURE 8.5 – Insertion de règle

Le modèle de requête JSON pour l'insertion de **politique** : figure 8.6
→ *@POST, @Path("postPolicyPath/policyName"), Response InsertPolicies(String policyAtt)*
Réponses possibles :

- http 201 Created si la règle est créée.
- http 400 Bad Request si le contenu du POST n'est pas valide.

62. http ://www.json.org/

```
{
    "PolicyName" : anID,
    "RuleName":[ ruleId1, ruleId2, ...]
}
```

FIGURE 8.6 – Insertion de politique

- http 500 Server Error si une erreur imprévue est survenue pendant la création de la règle.

Mise à jour et suppression

La suppression se fait par une requête http contenant l'identifiant (nom) du composant règle/politique à supprimer.
→ *@DELETE, @Path("deleteRulePath||deletePolicyPath /ruleName||policyName")* Par exemple :
http ://adresseIP/SaaS/resources/authzStre/deleteRulePath/Rules1

La mise à jour ce fait également via un appel REST avec les mêmes attributs du POST(figure 8.5 ou 8.6).
→ *@PUT, @Path("PutPath||PutPolicyPath /ruleName||policyName")*

Réponses possibles :

- http 200 OK si la règle est mise à jour.
- http 400 Bad Request si le contenu du POST n'est pas valide.
- http 404 Not Found si aucune règle n'est trouvée pour l'ID *"ruleId"*
- http 500 Server Error si une erreur imprévue est survenue pendant la mise à jour de la règle.

Dec-Reasoner

Pour la définition des composants de contrôle d'accès, à savoir règle, politique et politcySet, nous avons choisi d'utiliser un langage formel basé sur la logique temporelle *Event-calculus*.Une documentation complète sur le raisonneur que nous avons utilisé est dans [146]. La procédure d'installation (compilation) de *Dec-Reasoner* est également décrite dans le fichier **README** [63]. *Dec-Reasoner* prend en entrée un *fichier.**e*** et comme illustré dans la figure8.9, il retourne tous les résultats possibles du raisonnement sous la forme de modèles. On utilise un script *Python run_DecReasoner.py* (figure 8.7 [198]) en tant qu'intermédiaire à travers lequel on passe les ***fichiers.e*** au raisonneur, pour obtenir ensuite les résultats du raisonnement générés par ce dernier sous forme de ***fichier.res***.

Ensuite, comme notre service de contrôle d'accès est écrit en *Java*, nous avons implanté la méthode *DecReasonerInvocation* qui établie la connexion avec le raisonneur en passant par *run_DecReasoner.py*. La méthode *DecReasonerInvocation* illustrée dans la figure 8.8 prend un seul paramètre : *chemin*, qui est le chemin du *fichier.e* à passer au raisonneur.

63. http://sourceforge.net/projects/decreasoner/files/

```python
#! /usr/bin/env python

import sys
import optparse
import os
sys.path.append("/home/ahmed/decreasoner")
import decreasoner

def usage() :
    print "Usage: python %s [-o <output_file>] <input_file>" % os.path.basename(sys.argv[0])

def main() :
    open('.lock', 'w').close()

    # Option Parser
    parser = optparse.OptionParser()
    parser.add_option("-o","--output",action="store",type="string",dest="outputFilename")
    (options, args) = parser.parse_args()

    # Make sure we have our mandatory argument (file)
    if len(args) != 1 :
        print 'You must specify one file to process.'
        usage()
        os.remove('.lock')
        sys.exit(1)

    # Process
    print "The file to process is",args[0]
    if options.outputFilename :
        print "You used the --output option with value",options.outputFilename
        decreasoner.decreasoner().run(args[0],options.outputFilename)
    else :
        decreasoner.decreasoner().run(args[0],args[0]+'.res')
        outputFilename=""

    os.remove('.lock')
    sys.exit(0)

try:
    if __name__ == '__main__' :
        main()
except:
    print "Caught an exception"
```

FIGURE 8.7 – Script python pour invoquer le raisonneur.

```java
import org.python.core.PySystemState;
import org.python.util.PythonInterpreter;

public void decreasonerInvocation(String chemin) {
        try {
            PythonInterpreter.initialize(System.getProperties(),
                    PySystemState.getBaseProperties(),
                    new String[]{"./lib/decreasoner/run_decreasoner.py", chemin});
            PySystemState pss = new PySystemState();
            pss.setCurrentWorkingDir("/home/ahmed/GlassFish_Server/glassfish/domains/domain1/decreasoner/");
            PythonInterpreter interp = new PythonInterpreter(null, pss);
            interp.execfile("run_decreasoner.py");
        } finally {
            System.out.println(">>> reasoning performed.");
        }
    }
```

FIGURE 8.8 – Méthode java pour le déclenchement du raisonnement.

Note : Dans la logique d'exécution du raisonneur *Dec-Reasonner*, si une entrée commence par une minuscule, il l'a considère comme variable, et si elle commence par une majuscule il l'a considère comme une constante (i.e. valeur de la variable). Par conséquent, il est impératif que toutes les entrées dans la base de données *MongoDB* commencent par une Majuscule, faute de quoi le résultat fourni par *Dec-Reasoner* sera faux !

```
model 1:
time 0
F_ConditionSatisfied(Rule1).
F_ConditionSatisfied(Rule2).
F_ConditionSatisfied(Rule3).
F_ConditionSatisfied(Rule4).
F_RuleEffectNOTpermitted(Rule1).
F_RuleEffectNOTpermitted(Rule2).
F_RuleEffectNOTpermitted(Rule3).
F_RuleEffectNOTpermitted(Rule4).
Happens(E_DontMatchRuleParametters(Rule1), 0).
Happens(E_DontMatchRuleParametters(Rule2), 0).
Happens(E_DontMatchRuleParametters(Rule3), 0).
Happens(E_DontMatchRuleParametters(Rule4), 0).
time 1
+F_TargetDoesntHolds(Rule1).
+F_TargetDoesntHolds(Rule2).
+F_TargetDoesntHolds(Rule3).
+F_TargetDoesntHolds(Rule4).
Happens(ERuleDoesNotApply(Rule1), 1).
Happens(ERuleDoesNotApply(Rule2), 1).
Happens(ERuleDoesNotApply(Rule3), 1).
Happens(ERuleDoesNotApply(Rule4), 1).
time 2
+F_RuleNotApplicable(Rule1).
+F_RuleNotApplicable(Rule2).
+F_RuleNotApplicable(Rule3).
+F_RuleNotApplicable(Rule4).
Happens(E_PolicyDoesNotApply(Policy1), 2).
Happens(E_PolicyDoesNotApply(Policy2), 2).
time 3
+F_policyNotApplicable(Policy1).
+F_policyNotApplicable(Policy2).
Happens(E_PolicyDoesNotApply(Policy1), 3).
Happens(E_PolicyDoesNotApply(Policy2), 3).
Happens(E_policysetDontApply(PolicySet1), 3).
time 4
+F_policySetNotApplicable(PolicySet1).
```

FIGURE 8.9 – Résultat du raisonnement

Génération automatique des patrons Event-Calculus

Event-calculus est un langage de prédicats, et il n'est pas évident pour des utilisateurs non expérimentés d'écrire des règles en langage formel. Pour cette raison, nous avons créé une méthode de génération automatique de patrons *Event-Calculus* et nous l'avons intégré dans notre service de décision *CDS*. Comme le montre l'architecture globale du service de contrôle d'accès figure 8.4 au niveau des étapes **2,2'** et **3**, nous récupérons les valeurs des attributs des règles et les politiques stockés au niveau de la base de données, et notre service génère automatiquement les patrons adéquats. Les patrons sont des *fichier.e* et nous avons deux types de patrons :

- **Les patrons génériques et permanents** : sont immuables et représentent l'implantation de notre modèle de contrôle d'accès, il s'agit des patrons : **RulesPatterns, PolicyPatterns, PolicySetPatterns, sort (les variables du modèle) et ordering (pour la gestion des conflits)**. Ces patrons sont écrits en dur et ne doivent pas être changés.
- **Les patrons temporaires et dynamiques** : sont les patrons que le service génère et utilise pour chaque requête. Plus de détails sur ce type de patron ci-dessous.

Les différents patrons temporaires sont (*class SaaSPatterns*) :

- **input** : contient les informations de la requête à savoir : l'utilisateur, la ressource demandée et l'action désirée.
  → *generateInputfile (String user, String resource, String action, String workingDir)*

- **rules** : contient les informations pour chaque règle dans la base de règles. Le service parcourt la base de règles et génère pour chaque règle son patron.
  → *generateRulesPatterns(ArrayList<String> rules, String rulesWorkingDir)*.
- **policy** : contient les informations pour chaque policy dans la base de policies. Le service parcourt la base de policies et génère pour chaque policy son patron.
  → *generatePoliciesPatterns(String policies, String policiesWorkingDir)*.
- **policySet** : Le patron final que le service passe au raisonneur. Ce patron contient un pointeur vers tous les autres patrons Event-Calculus précédemment mentionnés.
  → *generatePolicySetPatterns(StringpoliciesSet, Stringpolicies, ArrayList < String > rules, StringpoliciesSetWorkingDir, StringrulesWorkingDir, StringpoliciesWorkingDir)*.

Bien que les méthodes de génération de patrons sont différentes, le principe est un peu le même. Plus précisément, le service interroge la base de données, récupère les informations concernant les différents composants de contrôle d'accès (la règle, la politique, le policySet et la requête) sous forme de données en format *JSON*. Ensuite le service de contrôle d'accès les parcourt, met les valeurs dans des tableaux dynamiques et enfin utilise les valeurs du tableau pour remplir les parties dynamiques (variables) dans les patrons prédéfinis.

Une fois que le service de contrôle d'accès a créé le fichier *PolicySet.e* il l'envoie au raisonneur via la méthode *DecReasonerInvocation >> run_DecReasoner.py*, et un fichier *PolicySet.res* est généré en conséquence. Ce dernier contient le résultat du raisonneur *Dec-Reasoner*. Le service de contrôle d'accès lit le contenu du fichier *PolicySet.res* et récupère le résultat du raisonnement sur la requête par rapport aux règles de contrôle d'accès existantes dans la base *MongoDB* et retourne le résultat finale sous forme de **boolean**. Au final, le service supprime tout les fichiers temporaires qui ont été créés pour éviter qu'ils soient réutilisés pour une nouvelle requête, notamment dans le cas d'une mise à jour de la base de règles ou politiques avec effet opposé à celles qui ont été créées.

La classe principale qui se charge de la gestion de tout le processus de contrôle d'accès est ***class SaaSResource >> authorizations (...)*** . Elle est accessible via ***@Path("authz")***. Par exemple, une requête de la part de l'utilisateur *"Member"* pour l'action "*GET*" sur la ressource "*Com1*" ressemble a :

```
http://.../SaaS/resources/authz?subject=Member&resource=Com1&action=GET.
```

Dépendances

— jython-standalone-2.5.2
— mongo-java-driver-2.12.3

Évaluation des performances

Pour l'évaluation des performances du temps de traitement pour l'analyse et la vérification des politiques de sécurité au moyen du raisonneur *DEC-Reasoner*, nous avons procédé à des évaluations sur un ordinateur portable avec un processeur *Intel Core i7-2640M CPU @ 2.80GHz, 8GB RAM et système d'exploitation Ubuntu 14.04 LTS*.

Pour le raisonnement, nous avons utilisé la version 1.0 du *Dec-Reasoner* avec le solveur SAT *Relsat-2.02*. Avant la phase du raisonnement, le *Dec-Reasoner* se charge d'abord de l'encodage des modèles *Event-Calculus* en un problème SAT. Cependant, une des principales limites du raisonneur *Dec-Reasoner* est le temps utilisé pour l'encodage SAT qui pourrait limiter le passage à l'échelle [201]. En effet, le code du *Dec-Reasoner* a été modifié par [203] proposant une modification au processus d'encodage. Cette modification a débouché sur une considérable amélioration des performances en matière du temps d'encodage. Nous avons utilisé la fonction améliorée pour l'encodage SAT pour les évaluations entreprises dans le cadre de cette thèse.

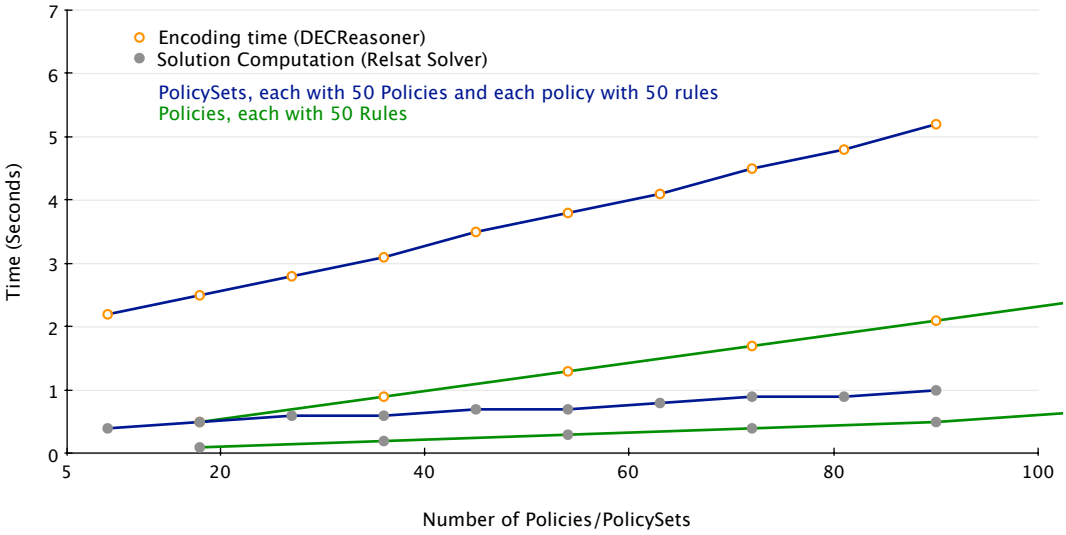

FIGURE 8.10 – Résultats d'évaluation des performances

En se référant au cadre principal (*i.e* partage de ressources) de l'exemple de motivation (**cf. chapitre Problématique et motivations**), nous avons testé plusieurs cas afin de mesurer les performances de notre approche. En premier lieu nous avons augmenté le nombre de règles, politiques et PolicySets. Néanmoins, les performances restent acceptables, de l'ordre de **0.2** secondes, jusqu'à un peu plus de 250 règles/politiques/PolicySets.

Les résultats du raisonnement sont illustrés dans la figure 8.10. L'axe Y indique le temps consommé, tandis que sur l'axe X sont affichées les cardinalités des ensembles politiques et PolicySets. Ainsi, nous avons considéré deux cas. Dans le premier cas nous augmentons le nombre de politiques, sachant que chaque chaque politique contient 100 règles. Les performances sont plutôt acceptables, car pour 180 politiques (180000 règles) le temps consommé pour l'encodage est **4.3** secondes et **1.4** secondes pour trouver la solution. Le second test introduit la notion de PolicySet. Ainsi on augmente le nombre de PolicySets, dont chacun compte 50 politiques contenant 50 règles. Le résultat reste également encourageant et rassurant par rapport aux performances de notre approche basée sur le raisonneur *Dec-Reasoner*. Car, même avec **90** PolicySet (4500 politiques et 225000 règles) le temps de de l'encodage SAT entre **5** à **6** secondes. Cependant, le temps nécessaire pour trouver la solution finale reste moins d'une seconde, et ce en dépit du fait que les expérimentations soient réalisées sur une machine

personnelle loin d'être du même ordre de performances de calcul qu'un serveur.

8.4 Audit et gouvernance des ressources collaboratives dans OpenPaaS

Dans cette partie, nous proposons une approche de suivi en temps réel du comportement des utilisateurs dans la plate-forme. Nous proposons également des mesures de prévention face aux comportements suspicieux des utilisateurs vis-à-vis des ressources collaboratives déployées dans une communauté. Cette supervision se fait par l'évaluation de la conduite de chaque utilisateur au sein de sa communauté pour évaluer la *confiance numérique* (*trust*). L'évaluation de la confiance entre dans le cadre de la gestion du risque pour la mise en place des politiques entreprises. Dans le cadre de la gestion du risque, le trust est un des paramètres qui permettent de décider si un acteur est autorisé (ou pas) à accéder à une ressource, et ce par rapport à un seuil de tolérance donné par l'entreprise propriétaire de la ressource collaborative.

Résumé de la solution proposée

L'analyse du comportement d'un acteur débouche sur la baisse ou l'augmentation de son *trust*. Ainsi, selon notre point de vue, le critère le plus approprié et efficace dans une plate-forme collaborative telle que *OpenPaaS* est le nombre de tentatives d'accès non-autorisés aux ressources collaboratives. En résumé, afin mesurer la réputation et ainsi le trust d'un acteur, nous nous intéressons au nombre de tentatives d'accès illégales qu'il tente.

Avant toutes choses, nous tenons à rappeler que la supervision inclut deux paramètres, à savoir la réputation et la confiance. La réputation est une estimation continue de la qualité du comportement d'un acteur. Il s'agit d'une valeur réelle qui peut changer à la fin de chaque session. En revanche, la confiance est un paramètre à valeur discrète calculée sur la base de l'évolution (positive ou négative) de la réputation d'un acteur. La confiance évolue d'une manière moins rapide que la réputation.

Comme illustré dans la figure 8.11, le module de supervision, intégré dans le service de contrôle d'accès, consulte le service de gouvernance *SuprvService*. Ce dernier reçoit en entrée les informations concernant les interactions de la session courante du sujet pour réévaluer sa réputation par rapport à un facteur de pénalité calculé en amont par le même service. Ces informations font ensuite l'objet de la mise à jour de l'historique d'expérience du sujet dans sa communauté.

Algorithme

Une vue détaillée sur nos procédures de calcul de la réputation ainsi que la confiance est illustrée dans la figure 8.12. Notre algorithme d'estimation de la réputation est basé sur la fonction *exponentialDecay* : $N(t) = N_0 e^{-\lambda}$. Cette fonction permet d'évaluer l'évolution aussi bien que la dégradation des résultats d'un phénomène à partir d'au moins deux échantillons extraits à des instants différents $\{t_0 ... t_n\}$. L'évolution ou la dégradation est déduite à partir de la valeur du *DecayFactor* λ. Si

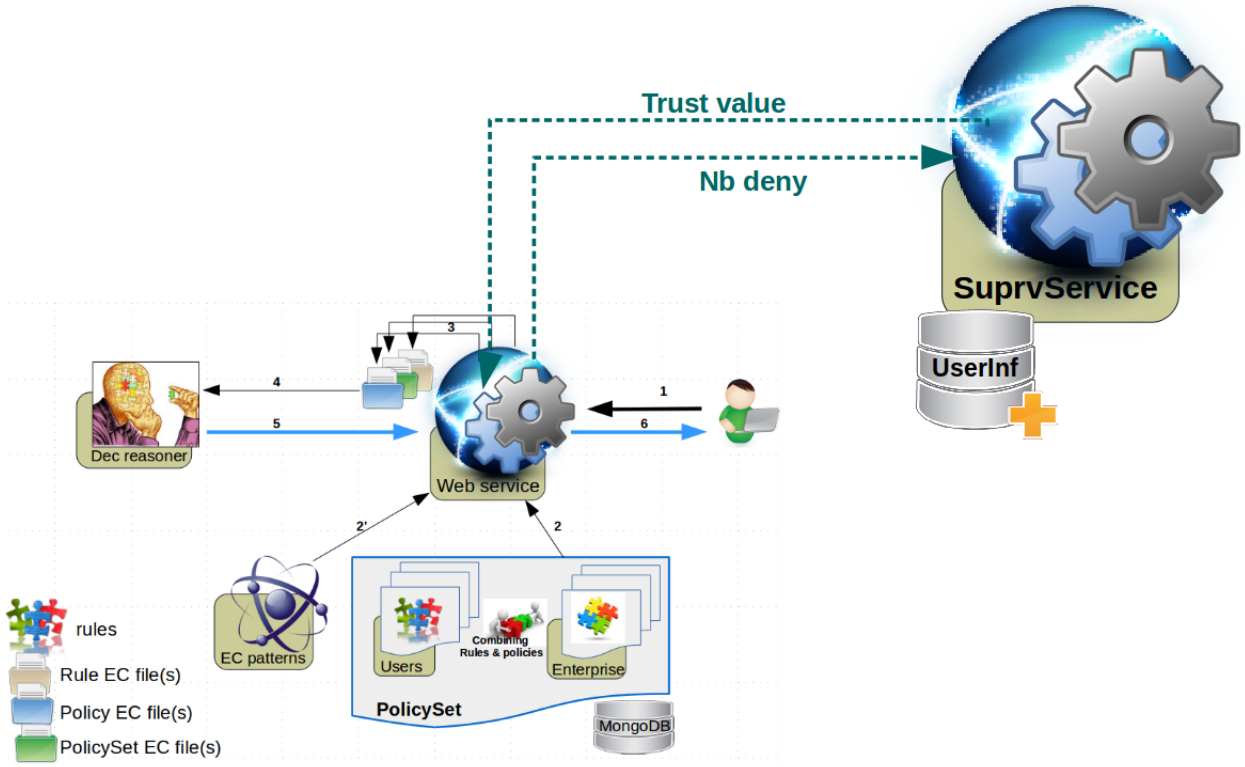

FIGURE 8.11 – Architecture du système de contrôle d'accès avec le module de gouvernance

$\lambda > 0 \rightarrow Growth$, si $\lambda < 0 \rightarrow Decay$.

La valeur de λ nous permet de projeter, au moyen d'une fonction exponentielle, le nombre de requêtes rejetées (*nbDeny*) sur la valeur de *trust* correspondante comme le montre la figure8.13. Sachant que, à chaque fin de session le *DecayFactor* (indice de trust) λ est recalculé en fonction de l'évolution (baisse/augmentation) de la *réputation*.

Implantation

Nous avons implanté en Java l'algorithme d'estimation de la réputation ainsi que celui du *trust* en *Java*, et nous les avons exposés en tant que service-Web REST. Nous utilisons une base de données pour le stockage des informations concernant le profil d'un sujet (*trust, decayFactor, historique*, *etc*). Comme le montre la figure 8.15, la méthode qui se charge du calcul de la valeur de trust *trComputing* prend en paramètre l'identifiant de l'utilisateur (*"requester"*), le nombre de requête(s) non-autorisée(s) *"nbDeny"* et le facteur de sévérité de la communauté en question. Ce dernier facteur représente le degré de pénalisation d'un utilisateur ayant dépassé le nombre de tentatives d'accès maximum (abus) autorisées par la communauté en question. Ce facteur de sévérité est un nombre réel positif $\in [0, 1]$, plus le nombre se rapproche de zéro (plus petit) plus l'utilisateur aura du mal à

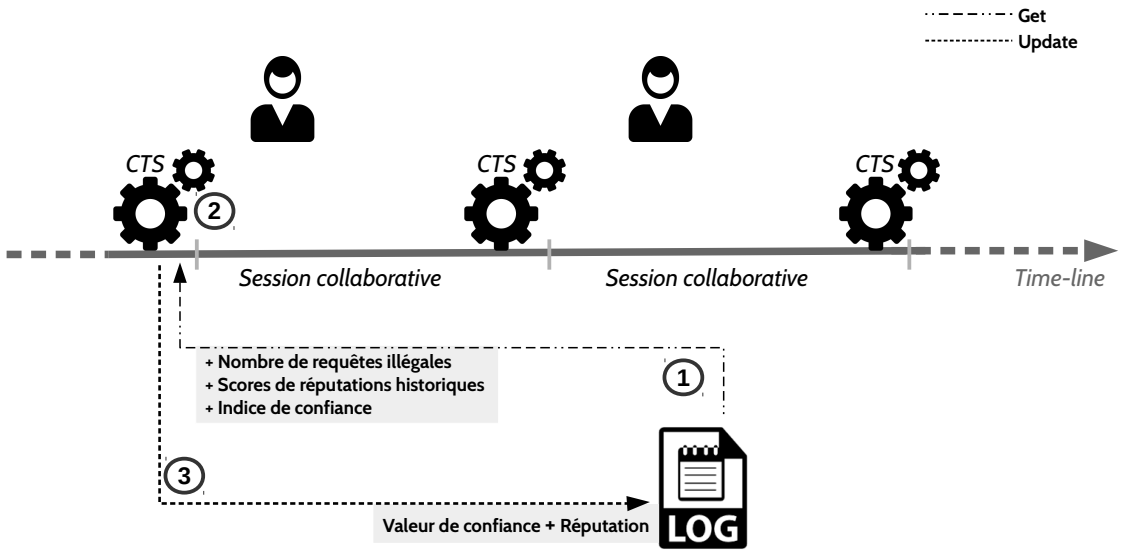

FIGURE 8.12 – Mécanisme d'évaluation de la réputation et la confiance vue détaillée.

faire évoluer sa valeur de confiance au sein de la communauté en question. La méthode *trComputing* est accessible via une requête http du type :

`http://.../AuditService/resources/trustComputing?subject=Member&nbDeny=2&comSeverity=0.01`.

La méthode *trComputing* est basée sur la méthode *evaluationTrust* (figure 8.16) qui prend en paramètres les mêmes paramètres que *trComputing* à savoir l'identifiant utilisateur (iduser), le nombre de refus qu'il a eu (nbDdeny) et la sévérité de la communauté dans laquelle il interagit (epsilon).

L'interaction avec la base de données utilisateurs *mongoDB* est gérée par la classe :
@Path("trustStre") public class ***TrustStore*** *{...}*

Dépendances

— javaee-api-7.0
— json
— mongo-java-driver
— activation-1.1
— javax.mail-1.5.0

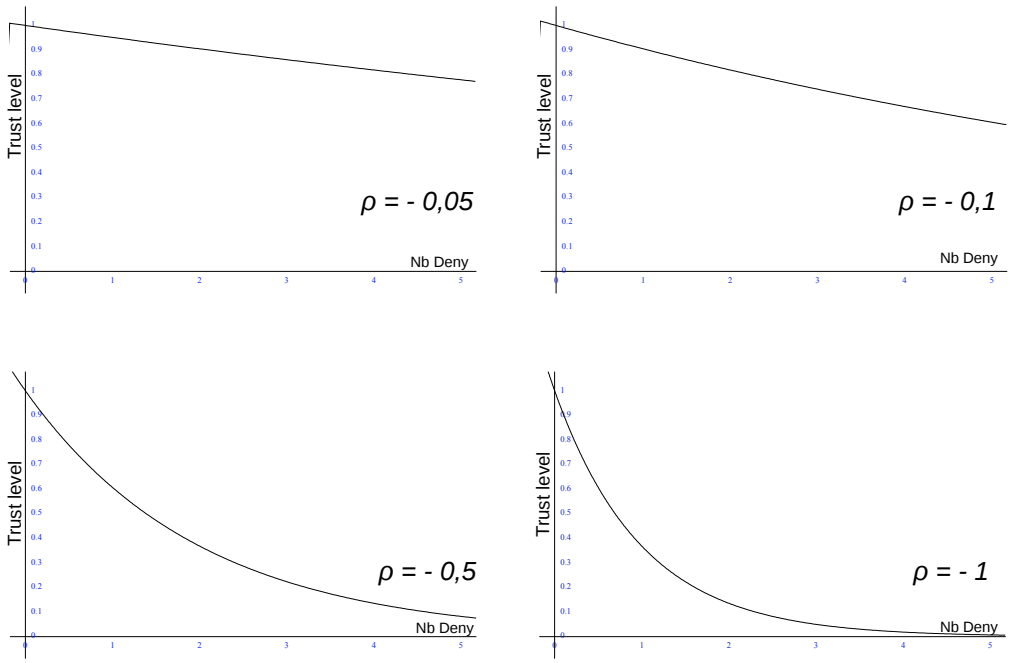

FIGURE 8.13 – Évolution du trust par rapport au DecayFacor

```
//killing session
    if (trust>=nbMaxDeny){
        HttpSession session = request.getSession(false);
        if (session != null) {
            session.invalidate();
        }
    }
```

FIGURE 8.14 – Interruption d'une session

```
@GET
public Response trComputing(@QueryParam("subject") String subject, @QueryParam("nbDeny") int nbDeny, @QueryParam("comSeverity")
    double comSeverity) throws IOException {

        double trVal = evaluationTrust(subject, nbDeny, comSeverity);

        return Response.status(Response.Status.OK).entity("trust value = " + trVal).build();
}
```

FIGURE 8.15 – Calcul du *trust*.

```
public static double evaluationTrust(long iduser, long idSession, int nbDeny, double eplison) throws UnknownHostException {
    double currentDecayFactor,
            lastDecayFactor = 0,
            currentTrValue = 0,
            pastTrValue = 0;

    double[] trEvaluationReferences = {-5, -1, -0.5, -0.1, -0.05};

    TrustStore trStore = new TrustStore();

    // Récupération de la dernière valeur du DecayFactor depuis la base de donnée
    lastDecayFactor = trStore.getDecayHist(iduser);

    ExpDecay expDec = new ExpDecay();

    //Calcul de la valeur de trust pour la session qui vient de se terminée
    if (nbDeny > 5) {
        currentTrValue = eplison;
        }
    else {
        currentTrValue = (float) expDec.calculateCurrentTrustValue(lastDecayFactor, nbDeny);
        }

    //Récupération de la dernière valeur de trust depuis la base de donnée: TrustHistory
    //Calcul de la moyenne des trust values passées
    pastTrValue = expDec.calculateAverageOfTrustHistory(trStore.getTrustHistory(iduser));

    //Calcul du DecayFactor propre à la session qui vient de se terminer (qui sera appliqué pour la session suivante)
    currentDecayFactor = (float) expDec.lambdaComputing(pastTrValue, currentTrValue);

    //Recupération de l'indice du Decay le plus proche par rapport à l'évolution du DecayFactor (past DecayFactors consideration)
    lastDecayFactor = expDec.calculateNewDecayFactor(lastDecayFactor + currentDecayFactor, trEvaluationReferences);

    //Ajout de la valeur de trust de la session à l'historique du trust de l'utilisateur
    // add decay value to the dataBase
    trStore.updateTrustHistory(iduser, currentTrValue);
    trStore.updateDecayHist(iduser, lastDecayFactor);
    return currentTrValue;
}
```

FIGURE 8.16 – Calcul du *trust*.

8.5 Conclusion

Dans ce chapitre nous avons abordé l'aspect technique de nos propositions des composants de sécurité conçu pour OpenPaaS RSE. En premier lieu nous avons détaillé l'implantation du mécanisme d'authentification interopérable et fédéré basé sur LemonLDAP-NG. Ensuite, nous avons détaillé le module de contrôle d'accès basé sur le modèle ABAC et la logique temporelle Event-Calculus. Ainsi, nous avons implanté notre modèle de sécurité sur la base d'une architecture XACML et un formalisme logique à l'aide de l'outil Dec-Reasoner pour l'analyse et la vérification des règles, politiques et PolicySets. Enfin, nous avons parlé de la mise en oeuvre du module de supervision et évaluation de la confiance numérique et la réputation.

```
/*
Example JSON request
http://adresseIP/AuditService/resources/trustStre/PostPath

msg:
{
"UserName" : "Ahmed",
"Role" : "Admin"
}
*/
@POST
@Path("PostPath")
public Response insertUserInfos(String msg) throws UnknownHostException {
    String errorReport = null;
    int status = 0;

    MongoClient mongoClient = new MongoClient();
    db = mongoClient.getDB(nameDataBase);

    JSONObject obj = new JSONObject(msg);
    String usrName = obj.getString("UserName");
    String Role = obj.getString("Role");

    if (!usrName.equals("") && !Role.equals("")) {
        errorReport = "Ok";
        status = 200;

        DBCollection CollectionUsers = db.getCollection("CollectionUsers");

        BasicDBObject userDoc1 = new BasicDBObject("UserName", usrName)
                .append("Role", Role);

        CollectionUsers.insert(userDoc1);

    } else {
        errorReport = "Bad Request";
        status = 400;
    }

    return Response.status(status).entity(errorReport).build();

}
```

FIGURE 8.17 – Initialisation de la base de données (*mongoDB*) des utilisateurs (ajout d'utilisateur "Ahmed")

Chapitre 9

Conclusion et perspectives

Sommaire

9.1 Synthèse

Dans cette thèse nous avons abordé le contexte des environnements collaboratifs destinés à des plateformes de type *réseau social d'entreprises*. Plus précisément, nous avons eu pour cadre de nos travaux de recherche le projet *OpenPaaS* RSE[64], dans lequel nous avons été chargés de la partie sécurisation des ressources et interactions collaboratives. En effet, *OpenPaaS* est une plateforme collaborative de type PaaS qui permet à différentes entreprises de déployer et d'utiliser des ressources logicielles et documentaires. Notre travail de gestion de la sécurité dans le cadre *OpenPaaS* couvrent les aspects *Authentification* (gestion des identifications), *Autorisation* (gestion des accréditations) et *Audit* (gestion de la réputation et de la confiance), et ce vis–à-vis des identités numériques qui vont collaborer.

La **gestion de l'authentification** des identités numériques a été notre premier challenge dans l'environnement *OpenPaaS*. En effet, *OpenPaaS* RSE est basé sur la notion de *communauté* qui consiste en un cercle collaboratif regroupant différentes entreprises **hétérogènes** en matière de gestion des identités numériques. Avec cet aspect d'hétérogénéité, nous avons eu pour objectif d'avoir une base fédérée de gestion de l'authentification qui permet à chaque partenaire (entreprise) de préserver ses préférences en matière de protocole de gestion de l'authentification. Pour cela nous nous sommes basés sur l'outil *"**LemonLDAP-NG**"*, avec une extension de l'architecture de fédération *"Credential-Based"* [32]. Ainsi, chaque acteur peut rejoindre une communauté de collaboration en s'authentifiant une seule fois (*i.e.* authentifiant unique *SSO*) par le biais du fournisseur d'identité de son entreprise, et avoir par conséquent son identité validée auprès de toutes les entreprises faisant partie de la communauté en question.

64. http://www.open-paas.org/

Après la gestion des identités, nous nous sommes focalisés sur la partie la plus importante pour la sécurisation des ressources collaboratives dans un environnement ouvert tel qu'une communauté RSE, à savoir la **gestion des autorisations d'accès** aux ressources partagées. Sur cette partie nous avons été amenés à faire face à de nombreux défis dus aux propriétés des environnements collaboratifs RSE. Nous sommes dans un contexte social destiné à rendre plus fluides les échanges collaboratifs. Ainsi, le premier challenge est de préserver cette fluidité d'échanges offerte par l'utilisation des RSEs. Cependant, il s'agit d'un environnement hétérogène en matière de gestion des autorisations. Cela signifie que la sémantique de gestion des profils identitaires des acteurs de collaboration en matière d'accréditations peut différer d'une entreprise à une autre. Par conséquent, le défi était de permettre aux entreprises de définir des permissions de contrôle d'accès **flexibles** vis-à-vis d'acteurs externes.

Par ailleurs, nous avons constaté le fait qu'il peut y a voir plusieurs entités impliquées dans la gestion d'une ressource collaborative (*e.g.* des entreprises, des acteurs, *etc.*). Cela implique la possibilité de combiner plusieurs permissions vis-à-vis d'une même ressource. En outre, cette combinaison nécessite une vérification des permissions définies afin d'éviter d'éventuelles incohérences dans le modèle d'autorisation. Vu qu'il s'agit d'un environnement ubiquitaire, cette vérification doit par ailleurs être peu coûteuse en temps, en coût de traitement et en mémoire.

Pour répondre aux précédentes exigences, nous avons défini deux types de politiques de contrôle d'accès, à savoir les **politiques de partage** et les **politiques de contrôle**, et ce sur deux niveaux, respectivement, le niveau acteur et le niveau entreprise. Pour la conception du modèle de politiques de contrôle d'accès, nous nous sommes basés sur un modèle **ABAC**, ce qui nous a permis d'avoir une bonne flexibilité en matière d'attributs qui nous a permis de gérer l'interopérabilité pour les politiques interentreprises.

Pour le développement de notre mécanisme de contrôle d'accès ABAC, nous nous sommes inspirés du protocole **XACML**. Cependant, pour ce faire, nous avons opté pour un formalisme logique basé sur la logique temporelle du premier ordre *Event-Calculus*. En guise de moteur de raisonnement et vérification des cohérences des politiques de contrôle d'accès, nous avons utilisé le logiciel ***Dec-reasoner*** basé sur le solveur SAT *Relsat*. L'aspect temporel de la logique *Event-Calculus* nous a également permis de gérer des permissions temporaires auto-révocables, à savoir les **délégations** qui sont très utiles et importantes dans notre contexte RSE.

Notre conception du mécanisme de contrôle d'accès est en réalité une extension du modèle XACML. Cette extension concerne un module de gestion de la confiance des sujets de collaboration au sein de chaque communauté. Ce modèle est l'un des principaux fondements des politiques de contrôle définies au niveau des entreprises. Ces politiques de contrôle permettent aux entreprises de garder le contrôle sur l'accès à leurs ressources partagées par leurs acteurs (partage *user-centric*) dans le contexte RSE. Plus précisément, une politique (*i.e.* un ensemble de règles) d'entreprise se combine aux politiques de partage de ressources afin de permettre aux entreprises d'avoir un contrôle abstrait sur les éventuels changements qui se produisent au sein du contexte de la communauté. Ces changements concernent plus précisément la confidentialité des différentes ressources partagées par les acteurs et les comportements des acteurs (fraudes, usurpation d'identité, *etc.*). Pour cela, nous

avons proposé un modèle de gestion basé sur un mécanisme de **évaluation du risque**, dans lequel nous avons aligné les concepts fondamentaux de gestion du risque, à savoir la menace, l'impact et la vulnérabilité avec ceux du contrôle d'accès et du contextes RSE, à savoir la confiance numérique, le nombre d'autorisations/propriétaires et l'authenticité des identités respectivement.

Les politiques d'entreprises sont basées sur des paramètres dynamiques. Cela donne par conséquent un aspect **dynamique** à notre mécanisme de **contrôle d'accès**. En effet, pour l'évaluation de la menace d'une requête reçue, le mécanisme de gestion du risque se base sur une évaluation dynamique et continue de la confiance que reflète le sujet de la requête. C'est pourquoi nous avons conçu un mécanisme d'**évaluation de la confiance** basé sur une évaluation dynamique et continue de la réputation de chaque acteur de collaboration. L'évaluation de la réputation est quant à elle basée sur l'historique de comportement et les informations de session courante concernant les tentatives illégales d'accès d'un acteur.

Enfin, nous avons implanté et intégré les prototypes des trois composants de sécurité **AAA** [65] dans la plateforme *OpenPaaS*. La mise en œuvre de ces modules est basée sur des services web.

9.2 Perspectives

Nos contributions dans cette thèse pour le projet *OpenPaaS* RSE couvrent les aspects essentiels de la sécurité, à savoir l'authentification, l'autorisation et la supervision. Cependant, il reste quelques détails et améliorations que nous n'avons pas traités dans le cadre de cette thèse, donnant ainsi une ouverture à plusieurs perspectives. Les améliorations peuvent être apportées sur les trois axes de recherche (**A**uthentification, **A**utorisation et **A**udit) abordés dans cette thèse.

D'abord sur l'aspect authentification, il est possible de travailler sur la conception d'une méthode basée sur la cryptographie pour établir un lien robuste entre l'identifiant interne et les identifiants externes d'un sujet de collaboration. Cela permettra de renforcer davantage la certification de l'authenticité de l'identité et réduire le risque d'usurpation d'identité. Des améliorations niveau prototype d'authentification sont également envisageables. Car pour l'instant, nous n'avons testé notre modèle de fédération interopérable basé sur *LemonLDAP-NG* qu'avec le mécanisme *login/mot-de-passe*, il sera également intéressant de voir comment le système se comporte avec d'autres configurations en matière de protocole de gestion de l'authentification.

Ensuite, sur l'aspect autorisation il est possible d'étendre notre mécanisme de délégation et améliorer les performances en temps de traitement des politiques d'autorisation. En effet, le raisonneur *Dec-reasoner* est basé sur un solveur SAT qui consomme beaucoup de temps pour l'encodage (problème NP-complet). Par conséquent, vu le temps de traitement par le *Dec-reasoner*, il devient presque impossible de raisonner sur des intervalles temporels assez long (*e.g.* dizaines de minutes). Une solution peut être la technique de *mémoisation* qui consiste à enregistrer l'état d'un système à un instant donné et de le réutiliser par la suite plutôt que de le recalculer, *i.e.* récursivité. Cela permet d'optimiser le coût de traitement d'une fonction en suspendant son exécution et la relancer sur la base des

65. **A**uthentification, **A**utorisation et **A**udit

résultats obtenus aux précédents calculs. Par exemple, ce principe de récursivité est utilisé dans le calcul de la *suite de Fibonacci*. Dans notre contexte, la *mémoisation* de la fonction du raisonnement sur les prédicats *Event-Calculus* (*i.e.* l'encodage SAT) peut être basée sur l'arrivée d'une nouvelle requête pour déterminer l'état du système (*i.e.* les fluents) et le stocker. Ensuite, relancer le processus de raisonnement sur la base des états antérieurs du système, et ce à l'arrivée de chaque nouvelle requête d'un sujet de collaboration.

Dans le même cadre du contrôle d'accès, il est également possible d'envisager des permissions inter-communautés. Plus précisément, un acteur peut transférer (migrer) certaines de ses permissions valables depuis une communauté qu'il souhaite quitter vers une nouvelle communauté qu'il souhaite rejoindre. Par ailleurs, nous considérons les délégations externes entre acteurs uniquement sur un seul niveau. En d'autres termes, nous n'avons pas étudié dans notre proposition l'éventualité qu'un acteur délégué puisse re-déléguer ses droits délégués, *i.e.* agir en tant que délégataire sur des droits qui lui sont délégués. Une délégation multi-niveau peut être pratique notamment dans un large environnement tel qu'un RSE. Concernant les délégations internes, nous n'avons pas abordé les politiques d'habilitation et de qualification d'une entreprise vis-à-vis de ses acteurs internes.

Enfin, concernant la dernière partie d'audit et supervision nous avons omis le traitement de certains paramètres, et nous avons proposé qu'ils soient définis d'une manière subjective. En effet, pour cette partie nous nous sommes basés sur un mécanisme de gestion du risque dans lequel les paramètres concernés sont les vulnérabilités de l'environnement collaboratif et le seuil de tolérance de risque. Concernant le deuxième paramètre (*i.e.* le seuil), de nombreuses techniques peuvent être utilisées telles que, *Coast* [47], ou l'évaluation des dommages (*damages*) [154, 143], voire même des techniques de seuil dynamique et adaptatif [42, 41]. Quant au premier paramètre concernant les vulnérabilités, de nombreux paramètres peuvent s'avérer aussi importants que la fiabilité du mécanisme d'authentification, comme par exemple la profondeur maximale de redélégation autorisée au sein de la communauté en question. La délégation multi-niveau peut également changer l'impact, qui fait également partie des paramètres du risque, sur les ressources collaboratives. L'autre paramètre dans la gestion du risque est la menace qui est basée sur la confiance d'un acteur. En effet, l'évaluation de la confiance est fondée sur les données historiques d'interactions de l'acteur au sein d'une communauté. Sachant qu'un acteur peut faire partie de plusieurs communautés (indépendantes) en même temps, les informations nécessaires pour l'évaluation de la confiance peuvent être transférées d'une communauté à une autre. En outre, il serait également intéressant que les informations historiques d'interactions subissent un pré-traitement afin de mieux les adapter en vue d'une exploitation objective dans le processus d'évaluation de la confiance numérique.

Bibliographie

[1] Caractéristiques d'openpaas : Collaboratio paas. https://www.linagora.fr/openpaas.

[2] Educause. things you should know about federated identity management. http://net.educause.edu/ir/library/pdf/EST0903.pdf.

[3] EmpowerID. ws-trust and ws-federation. https://www.empowerid.com/learningcenter/standards/ws-trust-fed.

[4] Microsoft. understanding ws-federation. https://msdn.microsoft.com/en-us/library/bb498017.aspx#wsfedver1_topic3.

[5] XACML-Studio (XS). http://xacml-studio.sourceforge.net/. [Online ; accessed 16-November-2015].

[6] AS/NZS 4360 SET Risk Management, Australian/New Zealand Standards, 2004.

[7] The epSOS project : A european ehealth project. http://www.epsos.eu, 2009. [Online ; accessed 16-November-2015].

[8] The Nationwide Health Information Network (NHIN) : an American eHealth Project. http://healthit.hhs.gov/portal/server.pt, 2009. [Online ; accessed 16-November-2015].

[9] OAuth2.0. http://oauth.net/about/, 2015. [Online ; accessed 08-December-2015].

[10] Présentation d'OAuth. https://support.google.com/a/answer/61017?hl=fr, mars 7, 2014. [Online ; accessed 08-December-2015].

[11] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 15(4) :706–734, 1993.

[12] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 9–pp. IEEE, 2000.

[13] J.-R. Abrial. *Modeling in Event-B : system and software engineering*. Cambridge University Press, 2010.

[14] W. J. Adams and N. J. Davis. Toward a decentralized trust-based access control system for dynamic collaboration. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 317–324. IEEE, 2005.

[15] A. Ahmed Nacer, E. Goettelmann, S. Youcef, A. Tari, and C. Godart. Business process design by reusing business process fragments from the cloud. In *The 8th IEEE International Conference on Service Oriented Computing and Applications*, page 8. IEEE, 2015.

[16] G.-J. Ahn and R. Sandhu. The rsl99 language for role-based separation of duty constraints. In *Proceedings of the fourth ACM workshop on Role-based access control*, pages 43–54. ACM, 1999.

[17] A. A. Almutairi, M. I. Sarfraz, S. Basalamah, W. G. Aref, and A. Ghafoor. A distributed access control architecture for cloud computing. *IEEE software*, (2) :36–44, 2011.

[18] F. T. Alotaiby and J. X. Chen. A model for team-based access control (tmac 2004). In *Information Technology : Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, volume 1, pages 450–454. IEEE, 2004.

[19] R. Ausanka-Crues. Methods for access control : advances and limitations. *Harvey Mudd College*, 301, 2001.

[20] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona : an online social network with user-defined privacy. In *ACM SIGCOMM Computer Communication Review*, volume 39, pages 135–146. ACM, 2009.

[21] A. K. Bandara, E. C. Lupu, and A. Russo. Using event calculus to formalise policy specification and analysis. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 26–39. IEEE, 2003.

[22] E. Barka and R. Sandhu. Framework for role-based delegation models. In *Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference*, pages 168–176. IEEE, 2000.

[23] E. Barka and R. Sandhu. Role-based delegation model/hierarchical roles (rbdm1). In *Computer Security Applications Conference, 2004. 20th Annual*, pages 396–404. IEEE, 2004.

[24] E. Barka, R. Sandhu, et al. A role-based delegation model and some extensions. In *23rd National Information Systems Security Conference*, pages 396–404. Citeseer, 2000.

[25] D. E. Bell and L. J. La Padula. Secure computer system : Unified exposition and multics interpretation. Technical report, DTIC Document, 1976.

[26] M. Ben Ghorbel-Talbi, F. Cuppens, N. Cuppens-Boulahia, and A. Bouhoula. Managing delegation in access control models. In *Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on*, pages 744–751. IEEE, 2007.

[27] E. Bertino, P. A. Bonatti, and E. Ferrari. Trbac : A temporal role-based access control model. *ACM Transactions on Information and System Security (TISSEC)*, 4(3) :191–233, 2001.

[28] E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A logical framework for reasoning about access control models. *ACM Transactions on Information and System Security (TISSEC)*, 6(1) :71–127, 2003.

[29] R. Bhatia, J. Lobo, and M. Kohli. Policy evaluation for network management. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1107–1116. IEEE, 2000.

[30] K. J. Biba. Integrity considerations for secure computer systems. Technical report, DTIC Document, 1977.

[31] A. Bielenberg, L. Helm, A. Gentilucci, D. Stefanescu, and H. Zhang. The growth of diaspora-a decentralized online social network in the wild. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 13–18. IEEE, 2012.

[32] E. Birrell and F. B. Schneider. Federated identity management systems : A privacy-based characterization. 2012.

[33] B. Blakley. The emperor's old armor. In *Proceedings of the 1996 workshop on New security paradigms*, pages 2–16. ACM, 1996.

[34] M. Blaze, J. Feigenbaum, and A. D. Keromytis. Keynote : Trust management for public-key infrastructures. In *Security Protocols*, pages 59–63. Springer, 1999.

[35] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 164–173. IEEE, 1996.

[36] M. Blaze, J. Feigenbaum, and M. Strauss. Compliance checking in the policymaker trust management system. In *Financial cryptography*, pages 254–274. Springer, 1998.

[37] R. V. Boppana, H. Maynard, and J. Niu. Attribute-based access control models and implementation in cloud infrastructure as a service. 2014.

[38] A. Bouchami, E. Goettelmann, O. Perrin, and C. Godart. Enhancing access-control with risk-metrics for collaboration on social cloud-platforms. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20-22, 2015, Volume 1*, pages 864–871, 2015.

[39] A. Bouchami and O. Perrin. Access control framework within a collaborative paas platform. In *Enterprise Interoperability VI*, pages 465–476. Springer, 2014.

[40] A. Bouchami, O. Perrin, and E. Zahoor. Trust-based formal delegation framework for enterprise social networks. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20-22, 2015, Volume 1*, pages 127–134, 2015.

[41] M.-R. Bouguelia, Y. Belaïd, and A. Belaïd. Efficient active novel class detection for data stream classification. In *ICPR-International Conference on Pattern Recognition*, pages 2826–2831. IEEE, 2014.

[42] M.-R. Bouguelia, Y. Belaïd, and A. Belaïd. An adaptive streaming active learning strategy based on instance weighting. *Pattern Recognition Letters*, 70 :38–44, 2016.

[43] D. F. Brewer and M. J. Nash. The chinese wall security policy. In *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*, pages 206–214. IEEE, 1989.

[44] D. W. Britton and I. A. Brown. A security risk measurement for the radac model. Technical report, DTIC Document, 2007.

[45] J. Bryans. Reasoning about xacml policies using csp. In *SWS*, pages 28–35, 2005.

[46] S. Buchegger and J.-Y. Le Boudec. A robust reputation system for peer-to-peer and mobile ad-hoc networks. In *P2PEcon 2004*, number LCA-CONF-2004-009, 2004.

[47] V. Cahill. Using trust for secure collaboration in uncertain environments. 2003.

[48] Capterra. Top Social Networking Software Products. http://www.capterra.com/social-networking-software/, 2008-2015. [Online ; accessed 27-October-2015].

[49] B. Carminati, E. Ferrari, and A. Perego. Rule-based access control for social networks. In *On the Move to Meaningful Internet Systems 2006 : OTM 2006 Workshops*, pages 1734–1744. Springer, 2006.

[50] B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in web-based social networks. *ACM Transactions on Information and System Security (TISSEC)*, 13(1) :6, 2009.

[51] S. Chakraborty and I. Ray. Trustbac : integrating trust relationships into the rbac model for access control in open systems. In *Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 49–58. ACM, 2006.

[52] A. K. Chandra and D. Harel. Horn clause queries and generalizations. *The Journal of Logic Programming*, 2(1) :1–15, 1985.

[53] Y. Cheng, J. Park, and R. Sandhu. Relationship-based access control for online social networks : Beyond user-to-user relationships. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 646–655. IEEE, 2012.

[54] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. In *Security and Privacy, 1987 IEEE Symposium on*, pages 184–184. IEEE, 1987.

[55] W. Clocksin and C. S. Mellish. *Programming in PROLOG*. Springer Science & Business Media, 2003.

[56] M. Colombo, F. Martinelli, P. Mori, M. Petrocchi, and A. Vaccarelli. Fine grained access control with trust and reputation management for globus. In *On the Move to Meaningful Internet Systems 2007 : CoopIS, DOA, ODBASE, GADA, and IS*, pages 1505–1515. Springer, 2007.

[57] C. Connolly. Managing privacy in identity management–the way forward.

[58] J. Crampton and H. Khambhammettu. Delegation in role-based access control. *International Journal of Information Security*, 7(2) :123–136, 2008.

[59] M. L. Crossley. *The desk reference of statistical quality methods*. ASQ Quality Press, 2000.

[60] F. Cuppens and A. Miège. Administration model for or-bac. In *On The Move to Meaningful Internet Systems 2003 : OTM 2003 Workshops*, pages 754–768. Springer, 2003.

[61] F. Cuppens and A. Miège. Modelling contexts in the or-bac model. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 416–425. IEEE, 2003.

[62] L. A. Cutillo, R. Molva, and T. Strufe. Safebook : A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE*, 47(12) :94–101, 2009.

[63] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The ponder policy specification language. In *Policies for Distributed Systems and Networks*, pages 18–38. Springer, 2001.

[64] M. I. Daud, D. Sánchez, and A. Viejo. Ontology-based delegation of access control : An enhancement to the xacml delegation profile. In *Trust, Privacy and Security in Digital Business*, pages 18–29. Springer, 2015.

[65] C. De Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. Generic aaa architecture. Technical report, 2000.

[66] Y. Demchenko, C. De Laat, L. Gommans, and R. V. Buuren. Domain based access control model for distributed collaborative applications. In *e-Science and Grid Computing, 2006. e-Science'06. Second IEEE International Conference on*, pages 24–24. IEEE, 2006.

[67] N. Dimmock, J. Bacon, D. Ingram, and K. Moody. Risk models for trust-based access control (tbac). In *Trust Management*, pages 364–371. Springer, 2005.

[68] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody. Using trust and risk in role-based access control policies. In *Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 156–162. ACM, 2004.

[69] J. Domingo-Ferrer. A public-key protocol for social networks with private relationships. In *Modeling Decisions for Artificial Intelligence*, pages 373–379. Springer, 2007.

[70] D. R. dos Santos, C. M. Westphall, and C. B. Westphall. Risk-based dynamic access control for a highly scalable cloud federation. In *7th International Conference on Emerging Security Information Systems and Technologies*, pages 8–13, 2013.

[71] A. A. El Kalam. *Modèles et politiques de sécurité pour les domaines de la santé et des affaires sociales*. PhD thesis, Institut National Polytechnique de Toulouse-INPT, 2003.

[72] A. A. El Kalam and Y. Deswarte. Multi-orbac : A new access control model for distributed, heterogeneous and collaborative systems. In *8th IEEE International Symposium on Systems and Information Security*, 2006.

[73] W. Ellis and D. Hersh. Xacml 3.0 analysis. 2015.

[74] N. B. Ellison et al. Social network sites : Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1) :210–230, 2007.

[75] A. B. Fadhel, D. Bianculli, and L. Briand. A comprehensive modeling framework for role-based access control policies. *Journal of Systems and Software*, 2015.

[76] D. Ferraiolo, D. R. Kuhn, and R. Chandramouli. *Role-based access control*. Artech House, 2003.

[77] Fido authentication. http://www.fidoalliance.org//.

[78] B. S. Firozabadi, M. Sergot, and O. Bandmann. Using authority certificates to create management structures. In *Security Protocols*, pages 134–145. Springer, 2002.

[79] K. Gaaloul, H. A. Proper, and F. Charoy. Delegation protocols in human-centric workflows. In *Commerce and Enterprise Computing (CEC), 2011 IEEE 13th Conference on*, pages 219–224. IEEE, 2011.

[80] M. Gaedke, J. Meinecke, and M. Nussbaumer. A modeling approach to federated identity and access management. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1156–1157. ACM, 2005.

[81] D. Gambetta. Can we trust trust. *Trust : Making and breaking cooperative relations*, 2000 :213–237, 2000.

[82] M. Gasser and E. McDermott. An architecture for practical delegation in a distributed system. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pages 20–30. IEEE, 1990.

[83] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas. Flexible team-based access control using contexts. In *Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 21–27. ACM, 2001.

[84] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas. Flexible team-based access control using contexts. In *SACMAT*, pages 21–27, 2001.

[85] P. D. Giang, L. X. Hung, S. Lee, Y.-K. Lee, and H. Lee. A flexible trust-based access control mechanism for security and privacy enhancement in ubiquitous systems. In *Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference on*, pages 698–703. IEEE, 2007.

[86] H. Gladney. Access control for large collections. *ACM Transactions on Information Systems (TOIS)*, 15(2) :154–194, 1997.

[87] E. Goettelmann, N. Mayer, and C. Godart. Integrating security risk management into business process management for the cloud. In *Business Informatics (CBI), 2014 IEEE 16th Conference on*, volume 1, pages 86–93. IEEE, 2014.

[88] L. González-Manzano, A. I. González-Tablas, J. M. de Fuentes, and A. Ribagorda. Cooped : Co-owned personal data management. *Computers & Security*, 47 :41–65, 2014.

[89] M. Gregg. *CISSP Exam Cram 2*. Que Corp., 2005.

[90] P. Hallam-Baker. Security assertions markup language. *May*, 14 :1–24, 2001.

[91] W. Han and C. Lei. A survey on policy languages in network and security management. *Computer Networks*, 56(1) :477–489, 2012.

[92] D. Hardt. The oauth 2.0 authorization framework. 2012.

[93] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8) :461–471, 1976.

[94] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone. Guide to attribute based access control (abac) definition and considerations. *NIST Special Publication*, 800 :162, 2014.

[95] V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo. Attribute-based access control. *Computer*, (2) :85–88, 2015.

[96] W.-K. Huang and V. Atluri. Secureflow : a secure web-enabled workflow management system. In *Proceedings of the fourth ACM workshop on Role-based access control*, pages 83–94. ACM, 1999.

[97] D. Jackson. *Software Abstractions : logic, language, and analysis*. MIT press, 2012.

[98] S. Jajodia, P. Samarati, M. L. Sapino, and V. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems (TODS)*, 26(2) :214–260, 2001.

[99] S. Jajodia, P. Samarati, and V. Subrahmanian. A logical language for expressing authorizations. In *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, pages 31–42. IEEE, 1997.

[100] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *IEEE Symposium on Security and Privacy*, pages 31–42, 1997.

[101] X. Jin. *Attribute-based access control models and implementation in cloud infrastructure as a service*. THE UNIVERSITY OF TEXAS AT SAN ANTONIO, 2014.

[102] A. K. Jones, R. J. Lipton, and L. Snyder. A linear time algorithm for deciding security. In *Foundations of Computer Science, 1976., 17th Annual Symposium on*, pages 33–41. IEEE, 1976.

[103] A. Jøsang. Trust and reputation systems. In *Foundations of security analysis and design IV*, pages 209–245. Springer, 2007.

[104] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2) :618–644, 2007.

[105] J. B. Joshi and E. Bertino. Fine-grained role-based delegation in presence of the hybrid role hierarchy. In *Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 81–90. ACM, 2006.

[106] J. B. Joshi, E. Bertino, and A. Ghafoor. Temporal hierarchies and inheritance semantics for gtrbac. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 74–83. ACM, 2002.

[107] A. Jsang and R. Ismail. The beta reputation system. In *Proceedings of the 15th bled electronic commerce conference*, pages 41–55, 2002.

[108] L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 63–74. IEEE, 2003.

[109] L. Kagal, T. Finin, and Y. Peng. A delegation based model for distributed trust. In *Workshop on Autonomy, Delegation, and Control : Interacting with Autonomous Agents, International Joint Conferences on Artificial Intelligence*, 2001.

[110] P. Kaila. Oauth and openid 2.0. *From End-to-End to Trust-to-Trust*, page 18, 2008.

[111] A. A. E. Kalam, R. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin. Organization based access control. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 120–131. IEEE, 2003.

[112] S. Kandala, R. Sandhu, and V. Bhamidipati. An attribute based framework for risk-adaptive access control models. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pages 236–241. IEEE, 2011.

[113] D. E. Knuth. Backus normal form vs. backus naur form. *Communications of the ACM*, 7(12) :735–736, 1964.

[114] V. Kolovski, J. A. Hendler, and B. Parsia. Analyzing web access control policies. In *WWW*, pages 677–686, 2007.

[115] P. Konopacki, M. Frappier, and R. Laleau. Expressing access control policies with an event-based approach. In *CAiSE Workshops*, pages 607–621. Springer, 2011.

[116] R. Kowalski and M. Sergot. A logic-based calculus of events. In *Foundations of knowledge base management*, pages 23–55. Springer, 1989.

[117] D. R. Kuhn, E. J. Coyne, and T. R. Weil. Adding attributes to role-based access control. *Computer*, (6) :79–81, 2010.

[118] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze. Cloud federation. In *Proceedings of the 2nd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2011)*, 2011.

[119] B. W. Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1) :18–24, 1974.

[120] G. Lawton. Developing software online with platform-as-a-service technology. *Computer*, 41(6) :13–15, 2008.

[121] A. Lazouski, F. Martinelli, and P. Mori. Usage control in computer security : A survey. *Computer Science Review*, 4(2) :81–99, 2010.

[122] H. Lee, I. Jeun, and H. Jung. Criteria for evaluating the privacy protection level of identity management services. In *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on*, pages 155–160. IEEE, 2009.

[123] R. Lepro. Cardea : Dynamic access control in distributed systems. *System*, 13(13) :4–2, 2004.

[124] H. Levesque, F. Pirri, and R. Reiter. Foundations for the situation calculus. *Linköping Electronic Articles in Computer and Information Science*, 3(18), 1998.

[125] N. Li and J. C. Mitchell. Datalog with constraints : A foundation for trust management languages. In *Practical Aspects of Declarative Languages*, pages 58–73. Springer, 2003.

[126] N. Li and J. C. Mitchell. A role-based trust-management framework. In *null*, page 201. IEEE, 2003.

[127] Y. Liu. Trust-based access control for collaborative system. In *Computing, Communication, Control, and Management, 2008. CCCM'08. ISECS International Colloquium on*, volume 1, pages 444–448. IEEE, 2008.

[128] Z. Liu, W. Sun, and M. Alam. A flexible role-based delegation model with dynamic delegation role structure.

[129] lemonldap-ng references. http://lemonldap-ng.org/references.

[130] Clément OUDOT. lemonldap : :ng un websso libre. http://fr.slideshare.net/coudot/lemonldapng-un-websso-libre?from_search=19.

[131] J. Lobo, R. Bhatia, and S. A. Naqvi. A policy description language. In *AAAI/IAAI*, pages 291–298, 1999.

[132] Ldap synchronization connector wiki. http://lsc-project.org/wiki/about/start.

[133] J. Lu, D. Xu, L. Jin, J. Han, and H. Peng. On the complexity of role updating feasibility problem in rbac. *Information Processing Letters*, 114(11) :597–602, 2014.

[134] W. Luo and E. Rosen. Method and system for providing authorization, authentication, and accounting for a virtual private network, Dec. 23 2008. US Patent 7,469,294.

[135] K. Lyytinen and Y. Yoo. Ubiquitous computing. *Communications of the ACM*, 45(12) :63–96, 2002.

[136] M. P. Machulak, E. L. Maler, D. Catalano, and A. Van Moorsel. User-managed access to web resources. In *Proceedings of the 6th ACM workshop on Digital identity management*, pages 35–44. ACM, 2010.

[137] E. Maler et al. Assertions and protocols for the oasis security assertion markup language (saml). *OASIS, September*, 2003.

[138] M. Mankai and L. Logrippo. Access control policies : Modeling and validation. In *5th NOTERE Conference (Nouvelles Technologies de la Répartition)*, pages 85–91, 2005.

[139] M. Masi, R. Pugliese, and F. Tiezzi. Formalisation and implementation of the xacml access control mechanism. In *Engineering Secure Software and Systems*, pages 60–74. Springer, 2012.

[140] N. Mayer. *Model-based Management of Information System Security Risk*. PhD thesis, University of Namur, Apr. 2009.

[141] K. McNamara. Rational fictions : central bank independence and the social logic of delegation. *West european politics*, 25(1) :47–76, 2002.

[142] N. H. Minsky. The imposition of protocols over open distributed systems. *Software Engineering, IEEE Transactions on*, 17(2) :183–195, 1991.

[143] I. Molloy, L. Dickens, C. Morisset, P.-C. Cheng, J. Lobo, and A. Russo. Risk-based security decisions under uncertainty. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*, pages 157–168. ACM, 2012.

[144] G. Mori, F. Paternò, and C. Santoro. Ctte : support for developing and analyzing task models for interactive system design. *Software Engineering, IEEE Transactions on*, 28(8) :797–813, 2002.

[145] T. Moses et al. Extensible access control markup language (xacml) version 2.0. *Oasis Standard*, 200502, 2005.

[146] E. T. Mueller. Discrete event calculus reasoner documentation. *Software documentation, IBM Thomas J. Watson Research Center, PO Box*, 704, 2008.

[147] E. T. Mueller. *Commonsense reasoning*. Morgan Kaufmann, 2010.

[148] E. T. Mueller. *Commonsense Reasoning : An Event Calculus Based Approach*. Morgan Kaufmann, 2014.

[149] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 2431–2439. IEEE, 2002.

[150] R. Nasim and S. Buchegger. Xacml-based access control for decentralized online social networks. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 671–676. IEEE Computer Society, 2014.

[151] National Institute of Standards and Technology. Information Security - Guide for Conducting Risk Assessments, 2002.

[152] T. N. Nguyen, K. T. L. Thi, A. T. Dang, H. D. S. Van, and T. K. Dang. Towards a flexible framework to support a generalized extension of xacml for spatio-temporal rbac model with reasoning ability. In *ICCSA (5)*, 2013.

[153] Q. Ni and E. Bertino. xfacl : an extensible functional language for access control. In *SACMAT*, pages 61–72, 2011.

[154] Q. Ni, E. Bertino, and J. Lobo. Risk-based access control systems built on fuzzy inferences. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 250–260. ACM, 2010.

[155] OASIS. XACML3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-1-en.html#_Toc230521654, 2013. [Online ; accessed 28-September-2015].

[156] J. Park and R. Sandhu. The ucon abc usage control model. *ACM Transactions on Information and System Security (TISSEC)*, 7(1) :128–174, 2004.

[157] J. S. Park, R. Sandhu, and G.-J. Ahn. Role-based access control on the web. *ACM Transactions on Information and System Security (TISSEC)*, 4(1) :37–71, 2001.

[158] S. Poslad. *Ubiquitous computing : smart devices, environments and interactions*. John Wiley & Sons, 2011.

[159] S. Radack. Risk management framework : Helping organizations implement effective information security programs. *INS Whitepaper, Santa Clara, INS*, 2009.

[160] P. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo. An algebra for fine-grained integration of xacml policies. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 63–72. ACM, 2009.

[161] I. Ray and M. Toahchoodee. A spatio-temporal access control model supporting delegation for pervasive computing applications. In *Trust, Privacy and Security in Digital Business*, pages 48–58. Springer, 2008.

[162] D. Recordon and B. Fitzpatrick. Openid authentication 2.0-final, 2007.

[163] G. Richard. De la confiance. *article paru dans la revue L'enseignement philosophique, 50e année n5*, 2000, 2000.

[164] E. Rissanen. extensible access control markup language (xacml) version 3.0, 2013.

[165] C. Ruan and V. Varadharajan. Dynamic delegation framework for role based access control in distributed data management systems. *Distributed and Parallel Databases*, 32(2) :245–269, 2014.

[166] A. Russo, R. Miller, B. Nuseibeh, and J. Kramer. *An abductive approach for analysing event-based requirements specifications*. Springer, 2002.

[167] J. A. Russo and R. Yates. Time limited collaborative community role delegation policy, Aug. 19 2008. US Patent 7,415,498.

[168] M. B. Saidi and A. Marzouk. Multi-trust_orbac : Access control model for multi-organizational critical systems migrated to the cloud. 2013.

[169] R. Sandhu, D. Ferraiolo, and R. Kuhn. The nist model for role-based access control : towards a unified standard. In *ACM workshop on Role-based access control*, volume 2000, 2000.

[170] R. S. Sandhu. Expressive power of the schematic protection model. *Journal of Computer Security*, 1(1) :59–98, 1992.

[171] R. S. Sandhu. Lattice-based access control models. *Computer*, 26(11) :9–19, 1993.

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

[173] R. S. Sandhu and P. Samarati. Access control : principle and practice. *Communications Magazine, IEEE*, 32(9) :40–48, 1994.

[174] V. Sassone, K. Krukow, and M. Nielsen. Towards a formal framework for computational trust. In *Formal Methods for Components and Objects*, pages 175–184. Springer, 2007.

[175] L. Schwittmann, C. Boelmann, M. Wander, and T. Weis. Sonet–privacy and replication in federated online social networks. In *Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on*, pages 51–57. IEEE, 2013.

[176] L. Seitz, E. Rissanen, T. Sandholm, B. S. Firozabadi, and O. Mulmo. Policy administration control and delegation using xacml and delegent. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 49–54. IEEE Computer Society, 2005.

[177] S.-W. Seong, J. Seo, M. Nasielski, D. Sengupta, S. Hangal, S. K. Teh, R. Chu, B. Dodson, and M. S. Lam. Prpl : a decentralized social networking infrastructure. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services : Social Networks and Beyond*, page 8. ACM, 2010.

[178] R. A. Shaikh, K. Adi, and L. Logrippo. Dynamic risk-based decision methods for access control systems. *Computers & Security*, 31(4) :447–464, 2012.

[179] M. Shanahan. The event calculus explained. In *Artificial intelligence today*, pages 409–430. Springer, 1999.

[180] R. Sharma and A. Datta. Supernova : Super-peers based architecture for decentralized online social networks. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1–10. IEEE, 2012.

[181] S. S. Shim, G. Bhalla, and V. Pendyala. Federated identity management. *Computer*, 38(12) :120–122, 2005.

[182] W. W. Smari, P. Clemente, and J.-F. Lalande. An extended attribute based access control model with trust and privacy : Application to a collaborative crisis management system. *Future Generation Computer Systems*, 31 :147–168, 2014.

[183] W. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. Travos : Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2) :183–198, 2006.

[184] Q. N. T. Thi and T. K. Dang. X-strowl : A generalized extension of xacml for context-aware spatio-temporal rbac model with owl. In *Digital Information Management (ICDIM), 2012 Seventh International Conference on*, pages 253–258. IEEE, 2012.

[185] R. K. Thomas. Team-based access control (tmac) : a primitive for applying role-based access controls in collaborative environments. In *Proceedings of the second ACM workshop on Role-based access control*, pages 13–19. ACM, 1997.

[186] J. E. Tidswell and T. Jaeger. Integrated constraints and inheritance in dtac. In *Symposium on Access Control Models and Technologies : Proceedings of the fifth ACM workshop on Role-based access control*, volume 26, pages 93–102, 2000.

[187] J.-Y. Tigli, S. Lavirotte, G. Rey, V. Hourdin, and M. Riveill. Context-aware authorization in highly dynamic environments. *arXiv preprint arXiv :1102.5194*, 2011.

[188] H. F. Tipton and M. Krause. *Information security management handbook*. CRC Press, 2003.

[189] K. Toumi, C. Andrés, and A. Cavalli. Trust-orbac : A trust access control model in multi-organization environments. In *Information Systems Security*, pages 89–103. Springer, 2012.

[190] H. T. T. Truong. *A Contract-based and Trust-aware Collaboration Model*. PhD thesis, Université de Lorraine, 2012.

[191] P. Tsankov, S. Marinovic, M. T. Dashti, and D. Basin. *Decentralized composite access control*. Springer, 2014.

[192] S. University of Murcia (UMU). UMU-XACML-Edito version 1.3.2. http://umu-xacmleditor.sourceforge.net/, 2009. [Online ; accessed 16-November-2015].

[193] H. D. S. Van, T. A. Dang, and T. K. Dang. Supporting authorization reasoning based on role and resource hierarchies in an ontology-enriched xacml model. *International Journal of Computer and Communication Engineering*, 3(3) :155, 2014.

[194] Q. Wang and H. Jin. Quantified risk-adaptive access control for patient privacy protection in health information systems. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 406–410. ACM, 2011.

[195] W. Wang. Team-and-role-based organizational context and access control for cooperative hypermedia environments. In *Proceedings of the tenth ACM Conference on Hypertext and hypermedia : returning to our diverse roots : returning to our diverse roots*, pages 37–46. ACM, 1999.

[196] N. Yang, H. Barringer, and N. Zhang. A purpose-based access control model. In *Information Assurance and Security, 2007. IAS 2007. Third International Symposium on*, pages 143–148. IEEE, 2007.

[197] S. S. Yau, Y. Yao, and A. B. Buduru. An adaptable distributed trust management framework for large-scale secure service-based systems. *Computing*, 96(10) :925–949, 2014.

[198] E. Zahoor. *Gouvernance de service : aspects sécurité et données*. PhD thesis, Université Nancy II, 2011.

[199] E. Zahoor, O. Perrin, and A. Bouchami. CATT : A cloud based authorization framework with trust and temporal aspects. In *10th IEEE International Conference on Collaborative Computing : Networking, Applications and Worksharing, CollaborateCom 2014, Miami, Florida, USA, October 22-25, 2014*, pages 285–294, 2014.

[200] E. Zahoor, O. Perrin, and C. Godart. Disc : A declarative framework for self-healing web services composition. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 25–33. IEEE, 2010.

[201] E. Zahoor, O. Perrin, and C. Godart. Disc : A declarative framework for self-healing web services composition. In *ICWS*, 2010.

[202] E. Zahoor, O. Perrin, and C. Godart. Disc-set : Handling temporal and security aspects in the web services composition. In *8th IEEE European Conference on Web Services (ECOWS 2010), 1-3 December 2010, Ayia Napa, Cyprus*, pages 51–58, 2010.

[203] E. Zahoor, O. Perrin, and C. Godart. An event-based reasoning approach to web services monitoring. In *ICWS*, 2011.

[204] L. Zhang, G.-J. Ahn, and B.-T. Chu. A role-based delegation framework for healthcare information systems. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 125–134. ACM, 2002.

[205] L. Zhang, G.-J. Ahn, and B.-T. Chu. A rule-based framework for role-based delegation and revocation. *ACM Transactions on Information and System Security (TISSEC)*, 6(3) :404–441, 2003.