Un système multi-agents basé sur une ontologie pour la détection d'intrusion dans les réseaux autonomiques

Laurent DEROCHE[1,2], Rémi LEHN[1,2], Henri BRIAND[1,2]

[1]LINA, 2, rue de la Houssinière, BP 92208, 44322 Nantes Cedex 03,
Equipe COnnaissance et Décision (COD)

[2]Ecole polytechnique de l'université de Nantes, la Chantrerie, rue Christian Pauc,
BP 50609, 44306 Nantes Cedex3

laurent.deroche@univ-nantes.fr, remi@fc.univ-nantes.fr, henri.briand@polytech.univ-nantes.fr

Résumé - Le calcul autonomique oriente aujourd'hui de nombreux travaux de recherche portant sur la gestion autonome des réseaux, et développe ainsi le concept de réseaux autonomiques. Parmi les objectifs d'auto-gestion du calcul autonomique, la sécurité, et plus particulièrement l'auto-protection des systèmes, nécessite des réponses proportionnées à des contraintes particulières afin de détecter des fonctionnements anormaux sur des réseaux hétérogènes et des systèmes de plus en plus distribués.

Abstract - The autonomic computing directs today numerous research on self-management of networks, and expands the concept of autonomic networks. Among the objectives of self-managing, security, and more particularly systems self-protection, requires proportionate responses to particular constraints to detect abnormal operations across heterogeneous networks and systems more and more distributed.

1 Introduction

L'autonomie dans les réseaux représente une nouvelle fonctionnalité essentielle qui est à l'origine de la future génération des réseaux autonomiques, réseaux qui auront la capacité de réagir à des évènements ou stimulis apparaissant dans leurs environnements. Les réseaux autonomiques reprennent les concepts d'auto-gestion édictés dans le calcul autonomique [1], ils doivent donc être capable de se configurer automatiquement, de chercher en permanence à améliorer leurs performances, de détecter, de diagnostiquer, de subvenir à des dysfonctionnements, et de se protéger d'attaques malveillantes. Sur ce dernier point il est nécessaire aujourd'hui de répondre à des problématiques de sécurité liées au nombre croissant d'éléments logiciels et matériels interagissant et à l'émergence de réseaux dynamiques et décentralisés. Pour cela plusieurs approches architecturales existent, notamment basées sur une gestion par politiques de sécurité et sur l'utilisation d'agents intelligents. Il est possible de combiner ces deux approches afin de concevoir une architecture multi-agents reposant sur une base de connaissances représentée sous la forme d'une ontologie. L'ontologie peut être utilisée afin de définir une modélisation conceptuelle de haut niveau de cette connaissance sur laquelle s'appuieront les agents qui, grâce à leur caractéristiques intrinsèques, pourront répondre de manière appropriée aux stimulis de l'environnement. L'utilisation d'une telle architecture se révèle alors propice au développement de systèmes de détection d'intrusion pour les réseaux autonomiques. Il est en effet possible de représenter de manière sémantique les tentatives d'intrusions et ainsi grâce aux capacités de réaction, d'anticipation et de sociabilité des agents, de prévenir et de faire avorter ces tentatives. Cet article se propose donc de présenter une architecture possible d'un système multi-agent de détection d'intrusion afin compléter le processus d'autonomisation des réseaux.

2 Etat de l'art

2.1 Le calcul autonomique

Le calcul autonomique est une initiative d'IBM lancée en 2001 avec la publication d'un manifeste [2] désignant la complexité croissante des systèmes comme un frein au progrès des technologies de l'information. Dans ce manifeste Paul Horn fait le constat qu'au taux de croissance actuelle il n'y aurait bientôt plus de personnes qualifiés pour maintenir les systèmes informatiques du

monde. En effet, les progrès importants réalisés dans le domaine des technologies de l'information ont considérablement complexifié l'informatique, ce qui a entraîné une nécessité de faire évoluer les caractéristiques et les capacités des éléments permettant de gérer ces ressources et leurs interconnexions. L'objectif de cette initiative est de simplifier la gestion des infrastructures en automatisant des processus et en injectant du savoir-faire directement au cœur des systèmes, les rendant ainsi plus autonomes. L'analogie peut être faite avec le système nerveux autonome qui permet à l'ensemble des fonctions vitales de notre corps (respiratoires, digestives et cardio-vasculaires) de fonctionner et de s'autoréguler sans que nous en ayons conscience, et sans que cela ne fasse intervenir de réflexion. Un système peut être appréhendé comme un ensemble de ressources informatique liées ensemble afin d'effectuer un certains nombre de fonctions préalablement définies. Un système peut donc être un serveur, une application, un équipement réseau, etc. Un tel système doit respecter un certain nombres de règles afin de pouvoir être considéré comme autonomique. Ces caractéristiques ont été définies en huit points clés :

1. le système autonomique doit disposer d'une connaissance détaillée de ses composants.
2. le système autonomique doit pouvoir se configurer et se reconfigurer sous l'effet de conditions variables et imprévisibles.
3. le système autonomique doit optimiser son fonctionnement de façon continue, afin de satisfaire les objectifs de haut niveau.
4. le système autonomique doit être capable de se restaurer suite à un dysfonctionnement, sans altération des données, ni défaut de traitement.
5. le système autonomique doit être capable de détecter, d'identifier et de se protéger afin de maintenir l'intégrité et la sécurité globale du système.
6. le système autonomique doit disposer d'une connaissance suffisante de son environnement afin de faciliter son adaptation.
7. le système autonomique doit pouvoir fonctionner dans un environnement hétérogène et implémenter des standards ouverts, ce ne peut donc pas être une solution propriétaire.
8. le système autonomique doit cacher sa complexité à l'utilisateur.

Pour la réalisation de ces objectifs, les systèmes autonomiques disposent d'une connaissance détaillée de leurs états internes et de leur environnement grâce à une supervision continue des changements pouvant intervenir. Cette supervision est effectuée grâce à une boucle de contrôle fermée qui va permettre la collecte d'évènements et donc déclencher les actions adéquates.

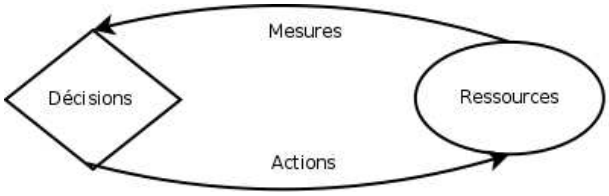

Figure 1: Boucle de contrôle

Le calcul autonomique vise donc à rendre des services performants et intelligents à des utilisateurs. Ces systèmes autonomiques doivent anticiper les besoins et permettre aux utilisateurs de se concentrer sur ce qu'ils souhaitent accomplir plutôt que sur la manière dont ils devraient configurer le système pour pouvoir y arriver. L'objectif final de l'évolution vers l'autonomie est d'aboutir à un système totalement autonome. L'évolution vers l'autonomie totale peut être décomposée en cinq niveaux :

Niveau 1 (de base, *basic*): il s'agit de la gestion traditionnelle, il y a de nombreuses sources de données systèmes générées. Ce niveau nécessite de nombreux personnels qualifiés pour l'administration et la maintenance. A ce niveau toutes les opérations sont effectuées manuellement.

Niveau 2 (gérée, *managed*): les donées sont consolidées au travers d'outils de gestion. Les personnels analysent et prennent des décisions. A ce niveau on note une amélioration de la productivité et meilleure connaissance du système.

Niveau 3 (prédictive, *predictive*): un système de contrôle permet une corrélation des informations et recommande des actions à prendre. Les personnels approuvent les recommandations et initient les actions à prendre. A ce niveau les prises de décision sont plus efficaces et les compétences nécessaires pour l'administrattion sont réduites.

Niveau 4 (adaptative, *adaptive*): un système de contrôle permet une corrélation des informations et prend les actions adaptées. A ce niveau les interactions humaines sont minimales.

Niveau 5 (autonomique, *autonomic*): les composants sont gérés dynamiquement par des politiques de haut niveau.

Pour atteindre ce but les acteurs de l'industrie doivent avoir une approche évolutionnaire et apporter des améliorations aux systèmes en place afin de ne pas avoir à remplacer l'intégralité des environnements liés aux technologies de l'information [3].

2.2 Rendre le réseau autonome

La gestion des réseaux est une activité traditionnellement contrôlée de manière manuelle par un ou plusieurs administrateurs qui sont en charge de la configuration, de la maintenance et de la sécurisation des environnements. Ces dernières années la convergence des services a changé la vue traditionnelle du réseau composé d'équipements homogènes pour celle d'un réseau englobant une multitude de technologies différentes (mobile/fixe, statique/ad-hoc, etc.), d'équipements hétérogènes et de services divers (temps-réel, qualité de service, etc.). Il apparaît alors que l'administration par l'homme de ces technologies devient de plus en plus complexe au vu du nombre de paramètres de configuration et de réglages que cela impose. Le concept de gestion autonome semble alors une solution à l'administration actuelle des réseaux afin qu'il puisse s'autogérer pour remédier à la complexité croissante et aux coûts excessifs de leur gestion. Pour rendre le réseau plus autonome plusieurs aspects importants sont à prendre en compte. Tout d'abord, il est nécessaire d'en décentraliser le contrôle et la gestion afin de permettre à ces composants d'être plus indépendants en terme de prise de décisions. Les entités composant le réseau doivent être capable de percevoir leur environnement et de réagir de manière adaptée aux évènements s'y produisant. Ces entités doivent être non seulement réactives mais également proactives, c'est à dire qu'elles doivent avoir des buts à atteindre qui dirigent leurs choix et leurs actions pour pouvoir mener une stratégie à long terme. De plus il est nécessaire que ces entités puissent coopérer ensemble tout en réalisant leurs objectifs propres, c'est à dire par exemple, pouvoir garantir une qualité de service de bout en bout à travers des réseaux hétérogènes. Pour réaliser ces buts une entité doit réaliser un certain nombre de plans de manière autonome, elle doit pour cela être capable de s'auto évaluer afin de pouvoir les modifier le cas échéant (dans le cadre d'une optimisation de ces performances).
Dans l'optique d'une auto organisation de ces réseaux, Une nouvelle architecture a été proposée, elle repose sur une architecture à quatre plans :

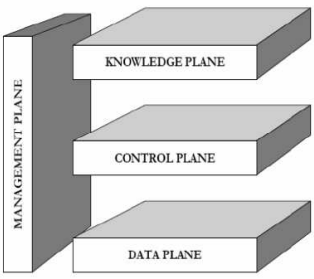

Fig. 2: Architecture à 4 plans

Le plan de connaissance est la pièce centrale de ce modèle architectural car il gère le réseau à travers le plan de contrôle. pour cela il choisit l'algorithme le plus approprié pour atteindre les objectifs fixés par l'opérateur. Il doit également décider de la configuration de tous les paramètres de l'algorithme et ensuite configurer le plan de contrôle qui lui-même gère la configuration du plan de données. Le plan de connaissance est bâti sur un système multi-agent cognitif, alors que le plan de contrôle est bâti sur un système d'agents réactifs.

2.2.1 Les agents intelligents

Il existe plusieurs définitions pour les agents, néanmoins elles s'accordent toutes sur le point qu'un agent est essentiellement un composant logiciel autonome se comportant comme un agent humain et fournissant des services à différents clients. Un agent peut percevoir son environnement au travers de capteurs (*sensors*), et peut réagir face à cet environnement au travers d'actionneurs (*actuators*).

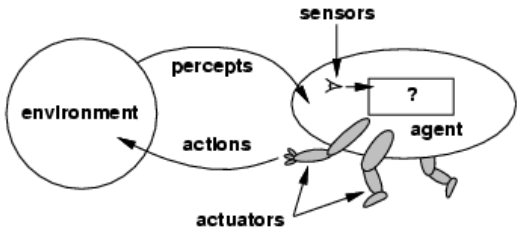

Fig. 3: Fonctionnement d'un agent

Il possède des caractéristiques qui lui sont propres :

- Autonome : il fonctionne sans intervention directe humaine ou autres , contrôle ses actions et son état interne.
- Situé : il agit sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement.
- Social : il coopère avec les humains ou d'autres agents afin de mener à bien les tâches lui étant dévolues.
- Réactif : il perçoit son environnement et réagit de manière appropriée pour effectuer les modifications nécessaires.
- Pro-actif : il est capable de se fixer des objectifs et de faire en sorte de les atteindre en prenant des initiatives.

De plus, si nécessaire, un agent peut être mobile, il a la capacité de se déplacer à travers les différents noeuds d'un réseau. Il peut être sincère, il ne peut fournir de fausses informations. Il peut être bienveillant, il essaie en permanence d'exécuter ce pour pour quoi il est programmé. Il peut être rationnel, il agit toujours pour réaliser ses objectifs, et il ne peut en aucun cas agir dans le but de les empêcher de se réaliser. Enfin il peut être adaptable, il apprend et s'adapte de lui-même à son environnement afin d'optimiser son fonctionnement et de mieux servir ses objectifs. Plusieurs aspects importants, étant des caractéristiques intrinsèques des agents, peuvent être définis pour rendre le réseau autonome [5]:

- la décentralisation qui doit permettre aux différents composants du réseau d'être plus indépendants et plus à même de décider des actions à entreprendre, et le cas échéant de faire appel à un administrateur humain ou à un autre composant autonome. En effet si le modèle client/serveur (utilisé dans le cadre d'une gestion humaine ou d'une gestion automatisée basée sur des politiques) permet d'avoir une meilleure vision du réseau dans sa globalité, cette solution reste lourde et non tolérante aux pannes.

- la réactivité qui doit permettre aux composants réseaux de s'adapter aux modifications de l'environnement en choisissant les mécanismes adaptés (adaptation des protocoles, modification des règles de sécurité de manière dynamique).

- la proactivité qui doit permettre une anticipation du comportement du réseau, en mettant en place des stratégies à long terme.

- la sociabilité qui doit permettre une coopération entre des réseaux et des équipements hétérogènes, par exemple pour la gestion d'une qualité de service de bout en bout.

- l'adaptabilité qui doit permettre l'optimisation des différents composants réseaux.

Les efforts initiaux concédés dans le cadre de systèmes basés sur des agents se sont concentrés sur la définition de modèles permettant l'émergence de plusieurs styles d'architectures, des plus simples (purement reactives) au plus complexes (basées sur le modèle *Belief-Desire-Intention*). Ces architectures d'agents intelligents peuvent être divisées en quatre catégories principales [4] :

- Les architectures basées sur la logique (*Logic-based/symbolic*) sont fondées sur des techniques traditionnelles basées sur la connaissance, dans lesquelles l'environnement est représenté de manière symbolique et manipulé en utilisant des mécanismes de raisonnement.

- Les architectures réactives (*Reactive*) où les décisions de l'agent sont prises en réponse à des stimulis de l'environnement (détectés grâce à des capteurs), les réponses à ces stimulis étant déclenchées par des actionneurs. Au contraire des architectures basées sur la logique aucun mécanisme de raisonnement complexe n'est utilisé et l'intelligence est distribuée entre de très nombreux agents.

- Les architectures BDI (*Belief, Desire, Intention*) sont basées sur le modèle "Croyance-Désir-Intention". Les croyances d'un agent sont les informations qu'il possède sur l'environnement et les autres agents qui existent dans le même environnement. Les désirs de l'agent représentent son état ainsi que l'état qu'il souhaiterait que l'environnement atteigne. Les intentions sont les actions que l'agent a décidé de mettre en oeuvre pour atteindre ses désirs.

- Les architectures en couches (*Layered*) ou hybride (*hybrid*) sont composées d'un ensemble de modules organisés dans une hiérarchie, chaque module étant soit une composante cognitive avec une représentation symbolique des connaissances et des capacités de raisonnement, soit une composante réactive.

Néanmoins, si les agents possèdent certaines capacités pour rendre autonome les réseaux, ils doivent être organisés en groupes afin de posséder toutes les capacités nécessaires à l'autonomie des réseaux, on parle alors de systèmes multi-agents. un système multi-agents consiste en un ensemble d'agents qui sont capables de communiquer entre eux, qui possèdent leurs propres ressources, qui perçoivent en partie leur environnement, qui ont une représentation partielle de leur environnement et qui ont un comportement dirigé vers la réalisation de leurs objectifs.

2.2.2 Les ontologies

Une ontologie est une spécification explicite d'une conceptualisation d'un domaine [7]. Une conceptualisation rend compte du sens des termes, et la spécification explicite fait des ontologies des objets syntaxiques. Lors de cette conceptualisation seules les propriétés essentielles sont définies car les propriétés étant en nombre indéfinies, on ne peut obtenir qu'une caractérisation partielle ou approximative du domaine. Selon Gruber [6], plusieurs critères sont à prendre en compte lors sa conception:

- La clarté : une ontologie doit fournir le sens attendu de termes définis, de manière indépendante du contexte. Lorsque c'est possible la définition doit être complète et documentée en langage naturel.

- La cohérence : l'ontologie ne doit autoriser que les inférences qui sont en accord avec les définitions afin de ne pas créer de contradictions.
- L'extensibilité : une ontologie doit être conçue afin d'anticiper l'utilisation du vocabulaire partagé. C'est à dire que l'ontologie doit être capable de définir de nouveaux termes pour des usages spéciaux, basés sur le vocabulaire existant, et ne nécessitant pas de modifications des définitions existantes.
- Une déformation d'encodage minimale : une déformation d'encodage apparaît lorsque les choix d'une représentation sont fait uniquement pour une commodité d'implémentation. Ceci doit être évité car les agents partageant la connaissance peuvent être implémentés dans différents systèmes de représentation et styles de représentation.
- Un engagement ontologique minimal : une ontologie doit faire aussi peu d'affirmations que possible sur ce qui a été modélisé afin de pouvoir instancier et spécialiser l'ontologie suivant les besoins.

Le but fondamentale d'une ontologie est d'améliorer la communication entre humains, entre humains et ordinateurs, entre ordinateurs et donc finalement d'aller vers un vocabulaire standardisé. Les ontologies peuvent être utilisées comme des composants de base pour construire un système autonomique. Elles peuvent être utilisées pour exécuter des analyses automatiques et continues basées sur des politiques de haut niveau définies pour détecter des menaces ou encore des dysfonctionnements. L'introduction d'une ontologie dans un système d'information (SI) vise à améliorer :

- Spécification : acquisition des connaissances pour aider à l'analyse des besoins et à la définition des spécifications.
- Réutilisation : composant réutilisable partagé par plusieurs logiciels.
- Fiabilité : réduire les coûts de maintenance en automatisant des vérifications de cohérence.
- Interopérabilité : format d'échange standardisé pour la coopération entre différents SI.

Il existe de nombreux langages permettant la conception d'ontologies. Parmi ceux-ci on distingue les langages basés sur la logique du premier ordre, comme NOTATION3, N-Triples et RDF (*Resource Description Framework*) , qui représentent les connaissances sous forme d'assertion (sujet, prédicat, objet), et les langages basés sur les logiques de description comme OWL (*Web Ontology Language*).

2.3 La sécurité dans les réseaux autonomiques

La sécurité informatique est l'ensemble des techniques qui assurent que les ressources (logicielles ou matérielles) du système d'information d'une entreprise ne seront utilisées que dans le cadre ou leur utilisation est prévue. Les objectifs de la sécurité informatique peuvent se décomposer en cinq parties principales :

- Confidentialité : protéger l'information de toute divulgation non autorisée, en utilisant un algorithme de chiffrement par exemple.
- Intégrité : garantir que l'information ne soit pas modifiée de manière malveillante par un tiers afin qu'elle reste fiable. Pour cela il est possible d'utiliser des algorithmes de hachage (SHA-1, MD5).
- Disponibilité : assurer la disponibilité des ressources lorsque leur usage est requis.
- Authentification : assurer l'authentification d'un tiers, en mettant par exemple en place des stratégies de contrôles d'accès par mots de passe.
- Non-répudiation : assurer qu'un tiers ne puisse répudier des actions commises ou qu'il est en train de commettre.

Les menaces sont multiples aujourd'hui, on peut citer les infections informatiques que sont les virus et les vers, les attaques de déni de service visant à nuire à la réputation ou au fonctionnement de certaines entreprises, les logiciels espions ou *spywares* visant à récupérer des informations personnelles de manière frauduleuse ou encore l'hameçonnage ou *Phishing* utilisé dans le but de perpétrer une usurpation d'identité. Pour toutes ces raisons les différents éléments autonomes constituant un système autonomique doivent être sécurisés afin de ne pas être compromis et détournés de leurs usages initiaux. La mise en place de cette sécurisation va de l'authentification des agents au chiffrement des communications inter-agents. Cet enjeu est critique, car en l'absence de contrôle humain sur le fonctionnement des éléments autonomes, il ne sera pas possible de se reposer sur l'esprit de déduction humain pour repérer un dysfonctionnement ou un comportement anormal. La sécurité informatique est donc un enjeu majeur pour la recherche aujourd'hui, avec l'apparition de nouvelles menaces chaque jour qui mettent à mal et éprouvent les protections des technologies existantes et émergeantes (démocratisation des accès sans fil). L'informatique autonomique doit permettre de rendre les systèmes plus sécurisé en autorisant le renforcement automatique et efficace de politiques de sécurité de haut niveau [8].

2.4 Travaux connexes

Durant les années 1990, les menaces comme les virus et les vers sont devenus un problème très préocupant pour la communauté informatique. Une des réponses proposées était le développement d'un système immunitaire artificiel inspiré du système immunitaire naturel [9]. Ce système sera précurseur de ce que sera le calcul autonomique et son architecture possède des aspects très interessant pour le développement d'éléments autonomes sécurisés. Le stockage distribué sécurisé ou SDS (*Secure Distributed Storage)* est une solution pour déployer de manière efficace des informations importantes sur de nombreux serveurs distincts de manière à augmenter la disponibilité et la fiabilité, et pour mettre en place des mécanismes d'auto-correction et d'auto-protection [10]. Dans le cadre de la détection d'intrusion des travaux ont été menés pour concevoir une architecture basée sur l'utilisation d'agents autonomes [11], qui consiste à faire travailler collectivement de multiples entités indépendantes. Plusieurs travaux de conception d'ontologies dans le cadre de la détection d'intrusion ont été menés. Jeffrey L. Undercoffer et *al* [12] ont définit une ontologie centrée sur la cible à l'aide du language DAML (*Darpa Agent Markup Language*). Cette ontologie permet une modélisation du domaine des attaques informatiques et facilite le processus de raisonnement pour pouvoir détecter et pallier aux intrusions malveillantes. Sa modélisation définit trois catégories, le composant système visé, les moyens mis en oeuvre, les conséquences de l'attaque, et la localisation de l'attaquant (local ou distant). Les travaux de Salvador Mandujano [13] ont permis la définition d'un système multi-agents basé sur une ontologie centrée sur l'attaquant, la modélisation ayant été réalisée en OWL. Cette architecture doit permettre non pas de protéger les ressources locales d'un système mais de prévenir l'utilisation de ces ressources pour compromettre des systèmes voisins. on peut également citer les travaux de Shao-Shin Hung et *al* [14], qui ont définit une ontologie centrée sur l'utilisateur.

3 Approche architecturale

L'approche architecturale présentée dans cet article repose sur l'utilisation d'un système multi-agents distribué et basé sur des politiques de haut niveau représentée grâce à une ontologie. L'objectif de ces recherches est de définir et de proposer une solution afin de concevoir une auto-protection pour les technologies liées aux systèmes d'informations et plus particulièrement dans le cadre de la détection d'intrusion. Un système de détection d'intrusion ou IDS (Intrusion Detection System) est un mécanisme écoutant le trafic réseau de manière furtive afin de repérer des activités anormales ou suspectes et permettant ainsi d'avoir une action de prévention sur les risques d'intrusion. Traditionnellement deux méthodes distinguent les différentes solutions proposées sur le marché, l'approche basée sur la connaissance et l'approche comportementale. L'approche basée sur la connaissance est la plus utilisée à ce jour, et part du principe que les attaques et vulnérabilités sont connues et que l'exploitation de toute vulnérabilité laisse une trace qu'il est possible de détecter. Pour fonctionner ce type de dispositif nécessite une connaissance exhaustive des traces d'attaques. L'approche comportementale est basée sur une modélisation du comportement normal du système à protéger, en l'absence d'intrusion. Par la suite toute déviation significative du comportement observé sera considérée comme une intrusion. Cette approche peut être vue comme paranoïaque. Dans notre approche nous nous proposons de définir une architecture permettant détecter des tentives d'intrusion connues en nous basant sur leurs signatures, et également d'anticiper ces intrusions en détectant des comportements anormaux, ces actions devant être gérées de manière autonome par le système. Notre auto-protection doit donc permettre à l'entité autonome de se défendre contre des attaques accidentelles et délibérées, et doit de plus éviter les intrusions et les accès non autorisés. Pour réaliser ces objectifs l'entité autonome doit intégrer un ensemble de fonctionnalités :

- Le système doit être autonome, il ne doit pas nécessiter l'intervention d'un opérateur pour éradiquer une menace.
- Le système doit être reconfigurable, afin de s'adapter et de lutter efficacement contre les menaces éventuelles.
- Le système d'auto-protection doit lui même être protégé, afin de ne pas pouvoir être détourné de ses fonctions.
- Le système doit pouvoir s'appuyer sur une base de connaissances, afin de pouvoir différencier les comportements normaux et anormaux. De plus la gestion de cette base doit être dynamique afin de suivre les évolutions du système.

Une telle architecture va permettre une configuration et une reconfiguration plus aisée des modules de sécurité, et va également faciliter la mise en place de nouvelles stratégies de sécurité.

3.1 Définition de l'ontologie

Notre ontologie modélise le domaine de la détection d'intrusion réseau. Le langage utilisé est le langage OWL-DL (*Web Ontology Language - Description logic*) et l'outil logiciel utilisé pour la modélisation est Protégé dans sa version 3.3.1 [15]. OWL est une recommandation du W3C [16] basée sur les langages OIL (*Ontology Inference Layer*) et DAML + OIL , permettant

de définir et d'instancier des ontologies. Une ontologie OWL est un graphe RDF et peut donc s'écrire avec des formes syntaxiques différentes. La signification d'une ontologie OWL est uniquement déterminée par le graphe RDF, l'utilisation de formes syntaxiques RDF/XML est admise tant que cela produit le même ensemble sous-jacent de triplets RDF. Un triplet RDF est une association sujet, prédicat, objet. Le sujet représente la ressource (URI, *Uniform Resource Locator*) à décrire, le prédicat un type de propriété applicable à cette ressource, et l'objet représente une donnée ou une autre ressource. La direction de l'arc est toujours pointée vers l'objet.

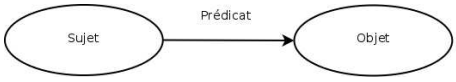

Fig. 4: Le triplet RDF

Une ontologie OWL peut contenir des descriptions de classes, de leurs propriétés et de leurs instances. La sémantique formelle OWL indique comment déduire ses conséquences logiques, c'est à dire comment déduire les faits non présents dans l'ontologie mais qui découlent de la sémantique. Une ontologie OWL est composée d'éléments de base comme des classes, des propriétés, des instances de classes et les relations entre ces instances. OWL se distingue du couple RDF/RDFS du fait qu'il intègre en plus des outils de comparaison des classes et des propriétés (équivalence, contraire, symétrie, etc). Le choix de OWL-DL pour notre modélisation a été guidé par le fait qu'il permet une forte expressivité tout en garantisssant la complétude des raisonnements (toutes les inférences sont calculables). De plus, de nombreux outils ont été développés afin de fournir des systèmes de raisonnement performants supportant les ontologies définit grâce à OWL-DL.
La Fig. 5 ci-dessous présente graphiquement la partie haute de notre ontologie, les ellipses représentant les classes et les sous-classes, et les arcs orientés les relations d'héritage et les propriétés d'objets. Dans cette partie haute nous avons définit les classes *HostSystem*, *HostComponent* et *Intrusion* (les classes *Router* et *Personal Computer* sont ici à titre d'exemples de sous-classes possibles de *HostSystem*). Il existe une relation *is victim of* entre la classe *HostSystem* et la classe *Intrusion* et une autre relation *has components* entre *HostSystem* et *HostComponent*.

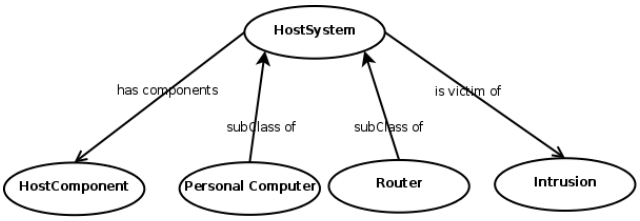

Fig. 5: Partie haute de l'ontologie

La Fig. 6 décrit la classe *HostComponent* et ses sous-classes. La classe *NetworkComponent* est composée de sous-classes descrivant les différentes interfaces réseaux. La classe *Protocol* permet la description de protocoles (IP, TCP, UDP). Il est à noter que les propriétés des sous-classes *Ip, Tcp,* et *Udp* ne représentent pas les caractéristiques des entêtes de ces protocoles, mais des valeurs relatives par exemple au nombre de connexions TCP simultanées autorisées ou encore aux versions du protocole IP autorisées. La classe *AgentComponent* permet la description des agents logiciels locaux et distants, en prenant en compte des caractéristiques comme le type de l'agent ou encore son GUID (*Globally Unique IDentifier*). Cette classe possède également une relation avec la classe *Intrusion* (non décrite sur la figure) et une relation avec la classe *Packet*.

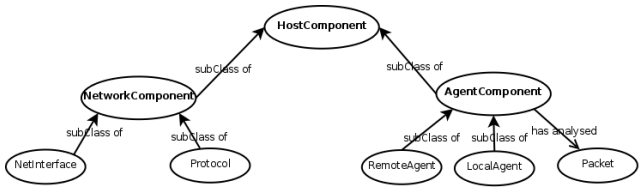

Fig. 6: Classe HostComponent

La Fig. 7 suivante décrit la classe *Packet* et ses sous-classes. La classe *Packet* permet de créer des instances de paquets contenant des informations liées aux entêtes IP, TCP et UDP. La classe *IpPacket* est composée de propriétés liées à l'entête IP (IP source, IP destination, informations de fragmentation, etc.), la classe *TcpPacket* est composée de propriétés liées à l'entête TCP (bits de contrôle, numéro d'acquittement, port TCP de destination, etc.), et la classe *UdpPacket* contient les propriétés liées à l'entête UDP (port UDP de destination et port UDP source).

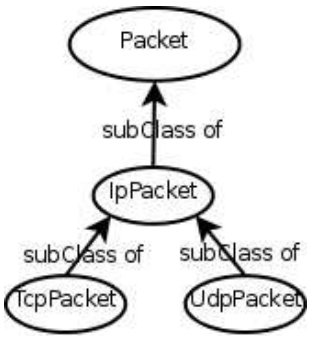

Fig. 7: Classe Packet

La Fig. 8 décrit la classe *Intrusion* et ses différentes sous-classes et relations. Elle permet la définition des différents types de menaces existantes, la sous-classe *DoS* en étant un exemple. Elle possède une relation avec la classe *Intruder* (permettant la description des éventuels intrus) et avec la classe *Signatures*. Elle possède des propriétés liées à l'heure de la tentative d'intrusion et aux contre-mesures possibles.

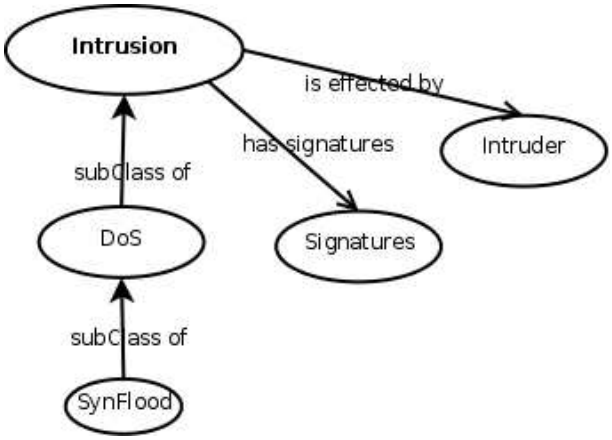

Fig. 8: Classe Intrusion

La Fig. 9 décrit la classe *Signatures* et ses différentes sous-classes. La classe *TrafficSignatures* caractérise le type de traffic généré par l'exécution d'une attaque. La classe *ProcessSignature* contient les propriétés d'exécution d'un programme et notamment son impact sur les ressources locales.

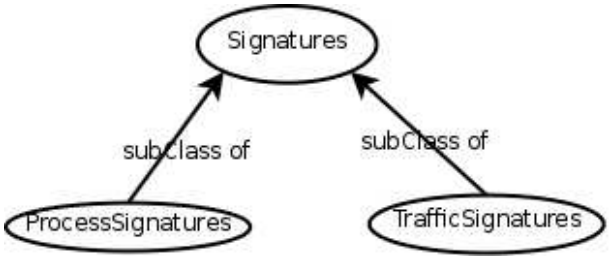

Fig. 9: Classe Signatures

3.2 Implémentation logicielle

Le schéma ci-dessous représente l'implémentation logicielle retenue pour ce projet de recherche.

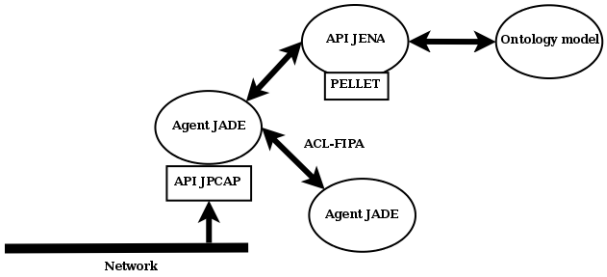

Fig. 10: Architecture logicielle

JPCAP[17] *Java network Packet CApture*, est une API (*Application Programming Interface*) Java open source composée d'un ensemble de classes fournissant une interface et des outils pour l'analyse réseau. JPCAP est basé sur l'API PCAP disponible en C. JPCAP est ici employée pour récupérer les paquets entrant sur une interface définie et pour en formater les informations. Les informations récupérées concernent les entêtes IP (version, longueur de l'entête IP, identification, flags, protocole), les entêtes TCP (ports TCP source et destination, numéros de séquence et d'acquittement, bits de contrôle) et les entêtes UDP (port UDP source et destination).

JADE[18] *Java Agent Development framework*, est un environnement de développement open source implémenté en Java et permettant de simplifier la création de systèmes multi-agents répondant aux spécifications FIPA[19] (*Foundation for Intelligent Physical Agents*). La FIPA a été initialement créee en 1996 afin de produire des standards logiciels pour les agents et les systèmes basés sur les agents et a été officiellement accepté en juin 2006 par l'IEEE (*Institute of Electrical and Electronics Engineers*). Jade est employé afin de créer les différents agents constituants le système, ce qui implique la modélisation de leurs comportements (Cycliques, séquentiels, etc.) et l'interaction entre les agents grâce à des messages de communication respectant les spécifications FIPA.

JENA[20] *Jena semantic web framework*, est un environnement Java open source autorisant la construction d'applications liées au web sémantique. JENA fournit un environnement de développement RDF, RDFS, OWL, SPARQL (*Simple Protocol and Rdf Query Language*), et propose plusieurs moteurs d'inférences internes, ainsi qu'une interface DIG (*DL Implementors Group*) afin de pouvoir utiliser un moteur d'inférence externe. JENA offre la possibilité de créer des ontologies, de manipuler des ontologies existantes en les im-

portant, et d'effectuer des requêtes sur ces ontologies. JENA permet également le stockage d'ontologies dans des bases de données relationnelles.

PELLET[21] *Pellet OWL-DL reasoner*, est un moteur d'inférence open source supportant OWL-DL. Il peut être utilisé en conjonction avec Jena et l'API OWL, et fournit une interface DIG. Il existe deux façons d'utiliser PELLET avec JENA, soit en utilisant directement l'interface de PELLET, soit en utilisant l'interface DIG de JENA, il est toutefois fortement conseillé d'utiliser l'interface de PELLET plus efficace et passant outre certaines limitations de l'interface DIG de JENA.

4 Perspectives et conclusions

Dans cet article nous proposons une approche distribuée et basée sur une définition sémantique des tentatives d'intrusion. Ce système de détection d'intrusion se base sur des comportements connus en utilisant des signatures caractéristiques de certaines attaques, mais l'objectif principal est de pouvoir adapter le comportement général du système par l'apprentissage de nouvelles attaques afin de pouvoir réagir aux attaques non connues. Ceci est rendu possible grâce à l'utilisation d'une ontologie et de ses capacités à pouvoir inférer de nouveaux comportements. De plus les propriétés de réutilisabilité et d'interopérabilité de ces ontologies sont essentielles afin de s'adapter aux problématiques de distribution des systèmes et d'évolution permanente des architectures réseaux. Dans cette optique de distribution des systèmes, l'utilisation d'agents intelligents pouvant interagir entre eux et se basant sur une ontologie, semble une bonne perspective pour assurer une sécurité efficiente et cohérente au sein des réseaux autonomiques .

References

[1] IBM, *An architectural blueprint for autonomic computing*, http://www-03.ibm.com/autonomic/pdfs/ACBP2_2004-10-04.pdf, 2004

[2] HORN P., *Autonomic computing : IBM's perspective on the state of information technology*, http://www.research.ibm.com/autonomic/manifesto/, 2001

[3] GANEK A. G., CORBI T. A., *The dawning of the autonomic computing era*, http://www.ibm.com/developerworks/autonomic/library/ac-summary/ac-ganek.html?\S_TACT=105AGX09&S_CMP=EDU, 2002

[4] BELLIFEMINE F., CAIRE G., GREENWOOD D., *Developping multi-agent systems with Jade*, WILEY, 2004

[5] KRIEF F., SALAUN M., *L'autonomie dans les réseaux*, LAVOISIER, 2006

[6] GRUBER T. R., *Toward principles for the design of ontologies used for knowledge sharing*, http://www.cise.ufl.edu/~jhammer/classes/6930/XML-FA02/papers/gruber93ontology.pdf, 1993

[7] GANDON F., *Ontologies informatiques*, http://interstices.info/display.jsp?id=c_17672, 2006

[8] CHESS D.M., PALMER C.C., WHITE S.R., *Security in an autonomic computing environment*, http://www.research.ibm.com/journal/sj/421/chess.pdf, 2003

[9] BURGESS M., *Computer immunology*, http://www.usenix.org/event/lisa98/full_papers/burgess/burgess.pdf, 1998

[10] GARAY A, GENNARO R., JUTLA C., RABIN T., *Secure distributed storage and retrieval*, http://citeseer.ist.psu.edu/garay97secure.html, 1997

[11] BALASUBRAMANANIYAN J., GARCIA-FERNANDEZ J., ISACOFF D., SPAFFORD E, ZAMBONI D., *An architecture for intrusion detection using autonomous agents*, http://citeseer.ist.psu.edu/balasubramaniyan98architecture.html, 1998

[12] UNDERCOFFER J., PINKSTON J, ANUPAM J., FININ T., *A target-centric ontology for intrusion detection*, http://www.cs.vu.nl/~heiner/IJCAI-03/Papers/Undercoffer.pdf, 2003

[13] MANDUJANO S., *An ontology-supported outbound intrusion detection system*, http://citeseer.ist.psu.edu/mandujano05ontologysupported.html, 2005

[14] HUNG S., LIU S.,*A User-centric Intrusion Detection System by Using Ontology Approach*, http://www.atlantis-press.com/publications/aisr/jcis-06/index_jcis.html?http%3A//www.atlantis-press.com/php/paper-details.php%3Fid%3D118,2006

[15] Protégé ontology editor, http://protege.stanford.edu/, 2007

[16] World Wide Web Consortium, *OWL Web Ontology Language Reference*, http://www.w3.org/TR/owl-ref/, 2004

[17] *Java network Packet CApture*, http://sourceforge.net/projects/jpcap, 2004

[18] *Java Agent Development framework*, http://jade.tilab.com/, 2007

[19] Foundation for Intelligent Physiacl Agents, http://www.fipa.org/, 2007

[20] *Jena semantic web framework*, http://jena.sourceforge.net/, 2007

[21] *Pellet OWL-DL reasoner*, http://pellet.owldl.com/, 2007