UNIVERSITE SAAD DAHLEB DE BLIDA

Faculté des sciences

Département d'informatique

MEMOIRE DE MASTER

En Informatique

Option : Sécurité des systemes d'informations

THÈME :

Mettre en oeuvre une politique d'identification, de detection et de sauvegarde des signatures des cyber-attaques dans une base de données ontologique pour son organisation

Réalisé par

YASRI Youcef

Encadrant et membres du jury

Mme.Soraya CHACHOUA

Mme.Yasmina GHEBGHOUB

Mr.Zakaria SAHNOUNE

Septembre 2022

Remerciements

Pour commencer je remercie dieu.

J'exprime aussi ma gratitude à mon encadrant et ma promotrice Madame Chachoua Soraya qui m'a fait l'honneur de diriger ce travail. je la remercie pour sa disponibilité et ses conseils qui m'ont été d'un apport considérable tant sur l'aspect scientifique que sur la méthode de recherche. Sans son appui et son expérience, rien n'aurait été possible.

Je remercie également les membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.

je remercie aussi tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail essentiellement Mehdi Mohamed Khaled Rabeh.

Je tient aussi a remercier mes parents qui m'ont toujours soutenue.

YASRI Youcef

Résumé

Le but de la détection d'intrusion est d'identifier les actions et les tentatives de contournement des politiques de sécurité afin de compromettre la confidentialité, l'intégrité ou la disponibilité d'une ressource et de déclencher des alertes lorsque une violation est détectée.

Les systèmes de détection d'intrusion (ou IDS : Intrusion detection System) utilisés aujourd'hui génèrent plusieurs alertes. Les informations qu'ils contiennent manquent de précision. Ces avertissements sont donc d'un intérêt limité pour un humain.

Avec notre protocole de détection, d'identification et de sauvegarde des intrusions dans une base de données ontologique, nous allons améliorer la précision des alertes et des blocages des systèmes de détection d'intrusions en diminuant les faux positifs et les faux négatifs.

Mots clés : Système de détection d'intrusion, IDS, sécurité, information, base de données, ontologie, protocole de sécurité, cyber sécurité, proxy.

Abstract

The goal of intrusion detection is to identify actions and attempts to circumvent security policies in order to compromise the confidentiality, integrity or availability of a resource and to trigger alerts when a violation is detected.

Intrusion detection systems (IDS) used today generate several alerts. The information they contain lacks precision. These warnings are therefore of limited interest to a human.

With our protocol for detecting, identifying and saving intrusions in an ontological database, we will improve the accuracy of alerts and blockages of intrusion detection systems by reducing false positives and false negatives.

Keywords: Intrusion detection system, IDS, security, information, database, ontology, security protocol, cyber security, proxy.

Liste des Abréviations

IDS	Intrusion Detection System
CVE	Common Vulnerabilities and Exposure
CERT	Computer emergency response team
CC	Coordination Center
NIST	National Institute of Standards and Technology
NIDS	Système de détection d'intrusions réseau
HIDS	Système de détection d'intrusion machine ou de type hôte
IPS	Intrusion Prevention System
NIPS	Network Intrusion Prevention System
TCP	Transmission Control Protocol
OSI	Open Systems Interconnection
HTTP	Hypertext Transfer Protocol
SIM	Security Information Manager
FTP	File Transfer Protocol
UDP	User Datagram Protocol
RPC	Remote procedure call
IP	Internet Protocol
IA	Intelligence Artificiel
OWL	Ontology Web Language
RDF	Resource Description Framework
URI	Uniforin Resource Identifier
OVM	Ontology Vulnerability Management
NVD	National Vulnerability Database
IOT	Internet of Things
MANET	Mobile ad hoc networks
IETF	Internet Engineering Task Force
DFO	Distributed Functions Ontology
SWRL	Semantic Web Rule Language
BGP	Border Gateway Protocol
TMC	Traffic Message Channel
RDS	Radio Data System
PMC	Perceptron Multicouche

Table des Matières

Liste des Figures

Liste des Tables

Introduction générale

Récemment, le monde connecté a connu une augmentation du nombre de vulnérabilités en matière de cybersécurité découvertes et signalées. Par exemple, Common Vulnerabilities and Exposure (CVE), une norme industrielle pour les identifiants de vulnérabilité et d'exposition, a publié 20 169 vulnérabilités entre le 1er janvier 2021 et le 31 décembre 2021[1]. Il s'agit du nombre le plus élevé de vulnérabilités divulguées au cours des dix dernières années. En outre, en 2014, CVE a adopté un nouveau format d'attribution des identifiants de vulnérabilité qui ne limite plus le nombre de CVE-ID à 10 000 par an.

Des recherches antérieures font état de plusieurs voies de divulgation des vulnérabilités[1][2]. Si le centre de coordination des équipes d'intervention en cas d'urgence informatique (CERT/CC)(Computer emergency response team / Coordination Center) est l'organisme officiel chargé de divulguer publiquement les vulnérabilités aux États-Unis, les découvreurs peuvent également mettre à disposition des informations sur les vulnérabilités par le biais des médias sociaux, des blogs et des wikis.

En particulier, les recherches suggèrent que les utilisateurs des médias sociaux divulguent et partagent souvent les vulnérabilités avant les sources officielles. Si la plupart des vulnérabilités sont exploitées entre le moment où elles sont découvertes et celui où un correctif est installé, les vulnérabilités graves telles que Heartbleed et VENOM, qui ont bénéficié d'une forte exposition sur les médias sociaux, ont été rapidement exploitées[3].

Bien que les informations sur les vulnérabilités soient divulguées et partagées par de multiples sources, dont le CERT/CC, les agences de sécurité, les listes de diffusion, ainsi que les médias sociaux, les abonnés à ces sources sont généralement des professionnels de la sécurité, des vendeurs ou des pirates. Dans la littérature existante, l'accent a été mis sur la gestion des vulnérabilités ; cependant, il y a un manque de recherche axée sur l'intégration des informations sur les vulnérabilités provenant de sources multiples qui pourraient être utilisées pour informer et protéger les utilisateurs communs des exploitations de vulnérabilités. En plus, l'intégration de l'intelligence des médias sociaux pour la gestion des vulnérabilités n'a pas reçu beaucoup d'attention. Quelques études récentes ont proposé des modèles conceptuels pour intégrer les informations des médias sociaux pour la cyberintelligence ; cependant, ces études ne tiennent pas compte des concepts de domaine de vulnérabilité tels que décrits dans le manuel du National Institute of Standards

[1] https://www.cvedetails.com/browse-by-date.php

and Technology (NIST)[4][5].

Sur la base de la littérature précédente, nous soutenons que la prévention des cyber-exploits nécessitera l'intégration d'informations sur les vulnérabilités provenant de multiples sources officielles et de renseignements sur les médias sociaux. Les informations sur les vulnérabilités peuvent être présentées dans un modèle ontologique structuré qui permettra aux analystes de sécurité d'évaluer la gravité des vulnérabilités et d'émettre des cyberalertes sur les vulnérabilités et les contre-mesures aux utilisateurs communs.

Dans ses recherches nous avons constituer une politique de détection, d'identification et de sauvegarde de cyberattaques dans une base de donnée ontologique afin d'augmenter l'efficacité des systèmes de détection d'intrusions à base de règles raisonnement sur l'ontologie du domaine.

Notre documents est structuré comme suit :

Introduction général dans la quelle nous avons défini la problématique et notre contribution à la solution.

Notre premier chapitre qui présente les systèmes de détection d'intrusions et le deuxième qui introduit les ontologies.

Nous avons ensuite montrer le fonctionnement des systèmes de détection d'intrusions a base ontologique dans le troisième chapitre.

Le quatrième et dernier chapitre décrit la solution que nous avons conçu.

Nous terminerons ce mémoire par une conclusion générale et quelques perspectives pour continuer à améliorer ce travail.

1. Chapitre I Les systèmes de détections d'intrusions

1.1 Introduction

Historiquement, internet a commencé comme un réseau privé qui réunissait des gouvernements, des institutions militaires et des chercheurs universitaires. Par conséquent, le besoin de protocoles de sécurités est faible on ne pensait pas a des virus, des vers, du spam, du phishing, des zombies, des logiciels espions, et des attaques de déni de services, etc. De plus, le coût d'une telle attaque n'est pas énorme. Progressivement, le réseau s'est ouvert sur le monde entier et sa taille a empêché la création d'un mécanisme totalement sécurisé. De plus, la commodité des programmes prêts à l'emploi et la prise en charge de l'automatisation des attaques permettent à quiconque de créer une agression rapide [6].

L'idée intuitive de la sécurité informatique est de restreindre l'accès à un système informatique. en toute sécurité, les données ne peuvent jamais être compromises car elles ne sont pas accessibles à un utilisateur non autorisé. Cependant, la sécurité parfaite est irréalisable et nous essayons toujours de prévenir, de détecter et de mettre fin à une attaque afin que ne puisse pas se répéter. La prévention permet de réduire les cas d'infractions. Pour ce faire, nous utilisons des techniques d'authentification, de cryptographie ou de masquage qui peuvent convaincre un attaquant potentiel que le test ne réussira pas. L'occurrence n'étant pas parfaite, la détection permet d'identifier les éléments en conflit avec les politiques de sécurité.

La détection d'intrusions a été introduite en 1980, qui a démontrer l'importance de la surveillance de la sécurité dans la détection d'éventuelles violations de la politique de sécurité du système, en particulier les violations de la confidentialité, de l'intégrité ou de la disponibilité du système [7]. En 1987, un modèle de détection d'intrusions a été publier[8]. En 1988, il y avait au moins trois prototypes, la recherche dans le domaine se développe alors, le nombre de prototypes augmente fortement[9]. L'objectif peut être différent: protéger des documents confidentiels, détecter de nouvelles attaques, respecter les restrictions légales telles que la confidentialité, signaler des intrusions ou des attaques, selon le schéma IDS, au responsable de la sécurité pour prendre les mesures appropriées, telles que la récupération du système à un état fonctionnel, etc. C'est une réaction nécessaire après la détection d'une intrusion. Les réponses peuvent être différentes, mais l'objectif est de réduire l'incidence des intrusions en améliorant le système de prévention et de détection et aussi de réparer les dommages causés par l'intrusion par d'éventuelles

poursuites judiciaires.

L'évaluation et la comparaison des systèmes d'analyse d'intrusion est un problème en soi en raison de la variabilité des sources de données potentielles et de la représentation des données utilisées principalement lors des tests.

1.2 Définitions

1.2.1 Intrusion

L'intrusion est toute utilisation d'un système informatique à des fins autres que celles prévues, généralement pour obtenir une autorisation de manière illégitime. Un attaquant est souvent considéré comme un étranger d'un système informatique qui peut le contrôler, mais les statistiques montrent que l'abus du détournement des ressources à l'espionnage industriel proviennent principalement de personnes internes ayant accès au système[9].L'entrée dans un système informatique a été décrite comme : "Tout ensemble d'actions qui tentent de compromettre l'intégrité, la confidentialité ou la disponibilité d'une ressource."[10] [11] Malgré les diverses formes d'intrusions, elles peuvent être regroupés en deux classes[12][13]:

- Intrusions connues : ces intrusions sont des attaques relativement bien définies qui exploitent généralement des failles connues dans le système cible;
- Intrusions inconnues ou anomalies : Ces intrusions sont considérées comme étrange du profil normal du système. Ils sont détectés lorsqu'un comportement anormal est détecté dans le système.

1.2.2 Détection d'intrusions

La détection d'intrusions consiste à analyser les informations recueillies par les contrôles de sécurité et à rechercher d'éventuelles attaques. C'est la capacité d'identifier les individus qui utilisent un système informatique sans autorisation et d'identifier ceux qui ont un accès légitime au système mais abusent de leurs privilèges.

Une intrusion se produit lorsque une victime subit des pertes au sens le plus large les conséquences de l'attaque. Ces attaques sont déclenchées par la présence de vulnérabilités systémiques que les attaquants exploitent pour atteindre leurs objectifs. Formellement, une attaque est une action prise par un ou plusieurs attaquants contre une ou plusieurs victimes, et la cible est atteignable. Cette action est une série d'événements qui affectent la sécurité

du système.

1.2.3 Système de détection d'intrusion(Intrusion Detection System-IDS)

Un système de détection d'intrusion est un dispositif qui surveille l'activité du réseau en tant que hôte donné pour détecter et répondre aux tentatives d'intrusion potentielles. Il est utilisé pour identifier les techniques d'analyse (analyse des ports, empreintes digitales), les tentatives de compromission des systèmes, les activités internes suspectes, les activités virales ou même pour vérifier les fichiers journaux. Un IDS peut déclencher deux types d'alertes:

- Faux positif: une alerte d'IDS qui n'est pas équivalente à une attaque réelle;
- Faux négatif: l'intrusion réel n'a pas été détecté par l'IDS.

La Figure 1 représente un modèle simplifie du système de détection d'intrusion dans le détecteur, qui analyse les informations du système contrôlé [14].

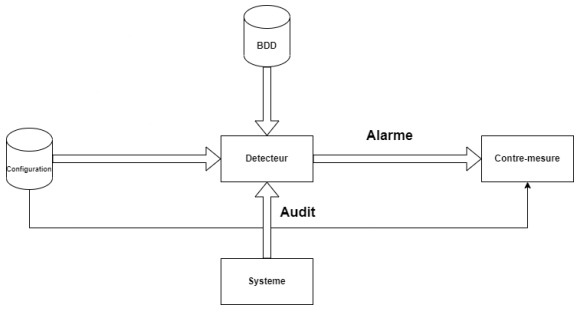


Figure 1. *Modèle simplifié d'un système de détection d'intrusions.*

Le détecteur analyse trois types d'informations: les informations à long terme liées aux techniques de détection (base de données de signatures), les informations de configuration qui déterminent l'état actuel du système et les informations d'audit qui décrivent l'événement dans le système. Trois méthodes ont été décrites pour évaluer l'efficacité des systèmes de détection d'intrusions [15][16], à savoir:

- Précision : la précision dans un système de détection d'intrusions déclare une activité légale comme nuisible. Ce critère équivaut à un faux positif.
- Performance : la performance du système de détection d'intrusion correspondent à la vitesse de traitement des événements. Si cette fréquence est faible,la détection en temps réel n'est pas possible.

- Complétude : la complétude est utilisé lorsque un système de détection d'intrusion ne détecte pas d'attaque. Cette mesure est la plus difficile car il est impossible d'avoir une connaissance globale des attaques. Cette échelle équivaut à un faux négatif.
  Les deux méthodes ci-dessous ont étés ajoutées[17][16]
- Tolérance aux pannes : Le système de détection d'intrusions lui-même doit résister aux attaques, notamment au déni de service. Ceci est important car de nombreux systèmes de détection d'intrusion fonctionnent sur du matériel ou des logiciels connus qui sont vulnérables aux attaques.
- Intervention précoce : Un système de détection d'intrusions doit être mis en place et les résultats de l'analyse diffusés dans les plus brefs délais afin que l'agent de sécurité puisse agir avant qu'un dommage ne se produise. Il ne s'agit pas seulement de calculer les performances et/ou le temps nécessaire pour traiter un événement, mais également du temps nécessaire pour se reproduire et traiter cet événement.

1.2.4 Composition d'un IDS

Il existe trois composants essentiels dans un IDS, à savoir:

1. Senseur[1] : Le senseur est responsable de la collecte d'informations du système, telles que des paquets d'un réseau ou des données de log;
2. Analyseur : L'analyseur reçoit l'ensemble des informations venant des senseurs. Il est responsable de les analyser et d'indiquer si une attaque est survenue ou non ainsi que son traitement. C'est essentiellement cette partie de l'IDS que ce document exposera.
3. Interface utilisateur : L'interface utilisateur permet aux utilisateurs de l'IDS de visualiser et/ou de définir le comportement du système.

1.3 Caractéristiques d'un système de détection d'intrusion

Un système de détection d'intrusions doit avoir les caractéristiques suivantes :

- Être en mesure d'effectuer une surveillance permanente et d'émettre une alarme en cas de détection;
- Fournir suffisamment d'informations pour réparer le système et de déterminer les dommages et la responsabilité de l'intrus;
- Être modulable et configurable pour s'adapter aux plates-formes et aux architectures

[1] `https://en.wikipedia.org/wiki/Wireless_sensor_network`

réseaux;

- Être en mesure d'assurer sa propre défense, comme supporter que tout ou une partie du système soit hors service;
- Avoir un faible taux de faux positifs;
- Être en mesure de tirer les leçons de son expérience et être fréquemment mis à jour avec de nouvelles signatures d'attaques;
- Être en mesure de gérer les informations envoyées par chacune des différentes machines et communiquer avec chacune d'entre elles;
- Être capable d'apporter une alerte automatique en cas d'attaques, mêmes coordonnées ou distribuées;
- Être en mesure de travailler avec d'autres outils, et notamment ceux de diagnostique de sécurité du système;
- Être en mesure de retrouver les premiers évènements d'attaque pour réparer correctement le système d'informations;
- Ne pas créer de vulnérabilités supplémentaires;
- Surveiller l'administrateur système.

  Lorsque le nombre de systèmes à superviser augmente et que, par conséquent, les attaques potentielles augmentent également, nous devons, avoir un système de détection d'intrusions qui a les caractéristiques suivantes :

- Il doit être capable de superviser un nombre important de stations tout en fournissant des résultats de manière rapide et précise.
- Il doit fournir « un service minimum en cas de crise » c'est à dire que si certains composants de système de détection d'intrusions cessent de fonctionner, les autres composants doivent être affectés le moins possible par cet état de dégradation.
- Il doit autoriser des reconfigurations et des installations de patchs d'une manière dynamique. Si un grand nombre de stations est supervisé, il devient pratiquement impossible de redémarrer le système de détection

1.4 Classification des systèmes de détection d'intrusions

Pour classifier les systèmes de détection d'intrusions, nous nous basons sur plusieurs variables. La principale différence retenue est l'approche utilisée, qui peut être soit comportementale, soit par scénarios. Nous verrons ensuite d'autres paramètres permettant de classer les différents systèmes de détection d'intrusions[16][18][19][20].

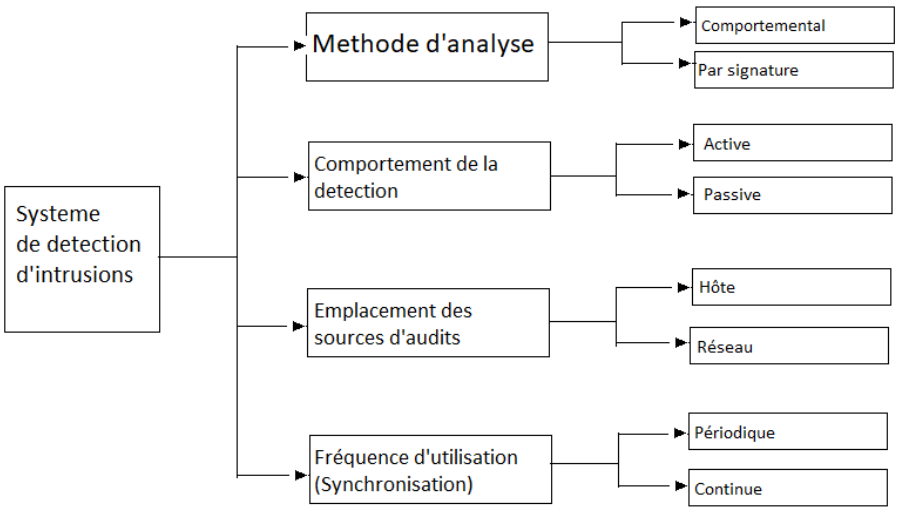


Figure 2. *Taxonomie des IDS*

1.5 La méthode de détection

Deux approches ont été proposées, l'approche comportementale (anomaly detection) et l'approche par scénario (misuse detection ou knowledge based detection). La première se base sur l'hypothèse que l'on peut définir un comportement «normal» de l'utilisateur et que toute déviation par rapport à celui-ci est potentiellement suspecte. La seconde s'appuie sur la connaissance des techniques employées par les attaquants qui permettent la définition des scénarios d'attaques en exploitant les traces d'audit de leur éventuelle survenue.

1.5.1 Approche comportementale

Une approche qui consiste à utiliser des méthodes basées sur l'hypothèse selon laquelle l'exploitation d'une vulnérabilité du système implique un usage anormal de celui-ci[7][8][21].

La détection d'anomalies consiste à définir, dans une première phase, un certain comportement du système, des utilisateurs, des applications, etc. considéré comme «normal». Dans une seconde phase, nous observons l'entité modélisée et tout écart par rapport au comportement de référence est signalé comme étant suspect. Cette approche recouvre en fait deux problèmes distincts, la définition du comportement «normal» souvent appelé «profil» d'une part, et la spécification des critères permettant d'évaluer le comportement observé par rapport à ce profil d'autre part.

Les différentes approches de détection d'anomalies se distinguent essentiellement par le

choix des entités modélisées dans le profil et l'interprétation qui est faite des divergences par rapport à ce profil.

Cette approche présente des avantages, tel que la capacité à détecter de nouvelles attaques.

Cependant, l'approche comportementale a des inconvénients, à savoir :

- Le choix des différents paramètres du modèle statistique est assez délicat et soumis à l'expérience de l'officier de sécurité;
- En cas de profonde modification de l'environnement du système cible, le modèle statistique déclenche un flot d'alarmes, du moins pendant une période transitoire;
- Un utilisateur peut changer lentement de comportement dans le but d'habituer le système à un comportement intrusif;
- Il est difficile de savoir si les observations faites pour un utilisateur particulier correspondent à des activités que nous voudrions prohiber;
- Pour un utilisateur au comportement erratique, toute activité est « normale ». Une attaque par déguisement sur son compte ne pourra pas être détectée;
- Il n'y a pas de prise en compte des tentatives de collision entre utilisateurs, alors même que cet aspect est très important, notamment dans le cas des réseaux.

1.5.2 Approche par scénario

Cette approche vise à détecter des signes de scénario d'attaques connues. Le principe commun à toutes les techniques de cette classe consiste à utiliser une base de données contenant des spécifications de scénario d'attaques à base de signatures. Le détecteur d'intrusions compare le comportement observé du système à ceux de la base et déclenche une alerte si ce comportement correspond à l'une des signatures existantes.

L'approche par scénarios s'appuie sur la connaissance des techniques utilisées par les attaquants pour déduire des scénarios déterminant.

Cette approche a des avantages et des inconvénients, parmi lesquels :

- Ce type de détecteur d'intrusions nécessite une maintenance active puisque par nature il ne peut détecter que les attaques dont les signatures sont dans sa base, cette base doit être régulièrement mise à jour en fonction de la découverte de nouvelles attaques.
- Aucune nouvelle attaque ne peut par définition être détectée, ce qui implique un taux plus élevé de faux négatifs.

- Le problème se pose essentiellement pour les attaques très récentes, dont les signatures n'ont pas encore pu être incluses dans la base.
- Il y a donc un besoin permanent de veille technologique et de maintenance, ce qui engendre un coût d'utilisation très élevé.
- La construction de cette base représente ainsi un problème à part entière et un système de détection d'intrusions de ce type doit s'accompagner d'outils efficaces de maintenance de la base.
- De manière générale, les détecteurs de scénarios se montrent fiables pour signaler les attaques référencées dans la base.
- Théoriquement, leur taux de faux positifs devrait rester très faible, car par définition une alerte n'est levée que dans le cas où la signature d'une attaque est observée.
- Cependant, pour des raisons de performance, les signatures sont souvent trop simples ce qui peut correspondre à des actions tout à fait légitimes.
- Le taux de faux positif reste donc élevé avec les outils existant aujourd'hui.
- De plus, une éventuelle connaissance de la base de signatures particulièrement dans le cas des patterns permet en principe à l'attaquant de construire précisément un scénario non détectable.
- Cela ne fait que renforcer encore l'exigence de maintenir régulièrement la base.
- Ce type de détecteur reste assez facile à mettre en œuvre, ne nécessitant pas de phase d'apprentissage ce qui élimine le risque de sur apprentissage ou de déformation volontaire du profil.

Il est indispensable d'hybrider 1'approche comportementale avec 1'approche par scénarios de manière à profiter des avantages de 1'une et de 1'autre.

1.6 Les types D'IDS

1.6.1 Introduction

Les intrusions et les attaques peuvent être commises soit du réseau soit directement de la machine et pour chaque cas nous avons un système adéquat

1.6.2 Système de détection d'intrusions réseau (NIDS)

Un NIDS écoute donc tout le trafic réseau, puis l'analyse et génère des alertes si des paquets semblent dangereux. Le but des NIDS est d'analyser de manière passive les flux en transit sur le réseau et détecter les intrusions en temps réel.

1.6.3 Système de détection d'intrusion machine ou de type hôte (HIDS)

Un HIDS se base sur une unique machine, n'analysant cette fois plus le trafic réseau, mais l'activité se passant sur cette machine. Il analyse en temps réel les flux relatifs à une machine ainsi que les journaux.

Un HIDS a besoin d'un système sûr pour vérifier l'intégrité des données. Si le système a été compromis par un pirate, le HIDS ne sera plus efficace. Pour parer à ces attaques, il existe des KIDS (Kernel Intrusion Detection System) et KIPS (Kernel Intrusion Prevention System) qui sont fortement liés au noyau.

1.6.4 Système de détection d'intrusion (hybride)

Généralement, il est utilisé dans un environnement décentralisé et il permet de réunir les informations de diverses systèmes d'écoutes placées sur le réseau. Leur appellation « hybride » provient du fait qu'ils sont capables de réunir aussi bien des informations provenant d'un système HIDS qu'un NIDS. L'exemple le plus connu dans le monde Open-Source est **Prelude**. Ce framework permet de stocker dans une base de données des alertes provenant de différents systèmes relativement variés. Utilisant **Snort** comme NIDS, et d'autres logiciels tels que **Samhain** en tant que HIDS, il permet de combiner des outils puissants tous ensemble pour permettre une visualisation centralisée des attaques.

1.6.5 Système de prévention d'intrusion(Intrusion Prevention System-IPS)

C'est un ensemble de composants logiciels et matériels dont la fonction principale est d'empêcher toute activité suspecte détectée de nuire au système. Contrairement aux IDS simples, les IPS sont des outils aux fonctions « actives » qui en plus de détecter une intrusion, tentent de la bloquer. Cependant, les IPS ne sont pas la solution parfaite comme on pourrait le penser. Plusieurs stratégies de prévention d'intrusions existent :

- Mémoire interne et protection des processus : surveille l'exécution des processus et les arrête si il considère que ces processus présentent un danger pour le système (buffer overflow).
- Interception de session : arrête une session TCP avec la commande TCP Reset : « RST ». Ceci est utilisé dans les NIPS (Network Intrusion Prevention System).
- Passerelle de détection d'intrusion : si un système NIPS est placé en tant que routeur,

il bloque le trafic sinon il envoie des messages à d'autres routeurs pour modifier leur liste d'accès.

Les IPS présentent de nombreux inconvénients :

- Un IPS bloque toute activité qui lui semble suspecte donc possibilité de bloquer un trafic inoffensif. Par exemple, un IPS peut détecter une tentative de déni de service alors qu'il s'agit simplement d'une période chargée en trafic. Les faux positifs sont donc très dangereux pour les IPS.
- Un pirate peut utiliser sa fonctionnalité de blocage pour mettre hors service un système. Prenons l'exemple d'un individu mal intentionné qui attaque un système protégé par un IPS, tout en usurpant son adresse IP. Si l'adresse IP usurpée est celle d'un nœud important du réseau (routeur, service Web, etc.), les conséquences seront catastrophiques. Pour pallier ce problème, de nombreux IPS disposent des « listes blanches », c'est-à-dire des listes d'adresses réseau qu'il ne faut en aucun cas bloquer.

1.6.6 Les firewall

Les firewalls ne sont pas des IDS, mais ils permettent également de stopper des attaques. Les firewalls sont basés sur des règles statiques afin de contrôler l'accès des flux. Ils sont utilisés en général au niveau des couches basses du modèle OSI[2] jusqu'au niveau 4[3], ce qui est insuffisant pour stopper une intrusion. Par exemple, lors de l'exploitation d'une faille d'un serveur Web, le flux HTTP sera autorisé par le firewall puisqu'il n'est pas capable de vérifier ce que contiennent les paquets.

1.6.7 Les techniques complémentaires

1. **Les scanners de vulnérabilités :** systèmes dont la fonction est d'énumérer les vulnérabilités présentes sur un système. Ces programmes utilisent une base de vulnérabilités connues.
2. **Les systèmes de leurre :** le but est de ralentir la progression d'un attaquant, en générant des fausses réponses telles que renvoyer une fausse bannière du serveur Web utilisé.

[2]Créé par l'Organisation internationale de normalisation, le modèle conceptuel OSI (Open Systems Interconnection) permet à divers systèmes de communication de communiquer à l'aide de protocoles standard. En clair, l'OSI constitue une norme permettant à différents systèmes informatiques de communiquer entre eux.

[3]Couche 4 dite transport : la couche transport sert de lien entre les couches orientées application et communication. A ce niveau du modèle OSI crée la connexion logique, soit le canal de transmission entre les systèmes communicants.

3. **Les systèmes de leurre et d'étude (Honeypots) :** le pirate est également leurré, mais en plus, toutes ses actions sont enregistrées. Elles seront ensuite étudiées afin de connaître les mécanismes d'intrusion utilisés par le pirate. Il sera plus facile d'offrir des protections par la suite.
4. **Les systèmes de corrélation et de gestion des intrusions (SIM - Security Information Manager) :** ils centralisent et corrèlent les informations de sécurité provenant de plusieurs sources (IDS, firewalls, routeurs, applications...). Les alertes sont ainsi plus faciles à analyser.
5. **Les systèmes distribués à tolérance d'intrusion :** l'information sensible est répartie à plusieurs endroits géographiques, mais des copies de fragments de l'information sont archivées sur différents sites pour assurer la disponibilité de l'information. Cependant, si un pirate arrive à s'introduire sur le système, il n'aura qu'une petite partie de l'information et celle-ci lui sera inutile.

1.7 Les types d'attaques

1.7.1 Les attaques réseau

Ces attaques sont souvent dues à une faille du protocole de sécurité réseau ou de son implémentions. Pour réaliser une telle attaque, une première étape est la récolte d'informations. Cette dernière peut se faire grâce au social engineering ou à des outils de scan. Le but de ces récoltes est par exemple l'obtention de la version des programmes, les services fonctionnels, les adresses IP ou les ports ouverts dans le système. Les scans peuvent se faire de plusieurs manières selon la volonté d'être plus ou moins visible lors du scan, et plus ou moins informé de la topologie du réseau. Il existe plusieurs méthodes telles que l'Open Scan, le Half-Open Scan, le Stealth Scan, le Sweep Scan, le Miscellaneous Scan[22].

- Open Scan : Open Scan est la technique de scan la plus simple. Cette technique utilise le protocole TCP et le drapeau SYN afin de détecter les ports TCP. Quand un port fermé est ciblé, la victime répond avec un drapeau RST. D'un autre côté, quand un port ouvert est détecté, la victime répond avec un drapeau ACK. L'avantage de cette technique est qu'elle peut réaliser son analyse d'une manière très simpliste sans nécessiter d'autres fonctionnalités ou privilèges. Cette technique simple est facilement détectée par un pare-feu[23].
- Half-Open Scan : (Scanner semi-ouvert) Le scan Half-Open, communément appelé le TCP SYN scan, est une méthode courante d'identification des ports. qui permet au scanner de rassembler des informations sur les ports ouverts sans terminer le

processus de prise de contact TCP. Puisque cette technique de scan ne crée jamais de session TCP, elle présente deux avantages. Premièrement, elle n'est pas enregistrée par les applications de destination. Deuxièmement, elle est moins stressante pour le service d'application parce qu'elle car elle n'oblige pas l'application à s'initialiser ou à allouer des ressources système. D'un autre côté, cette méthode souffre d'un inconvénient. Puisqu'il est nécessaire de créer de nouveaux paquets bruts qui ne respectent pas complètement la poignée de main TCP, le processus nécessite des privilèges élevés [24].

- Stealth Scan : (Scan furtifs) Les techniques d'analyse cybernétique susmentionnées utilisent seulement le drapeau SYN typique pour enquêter sur les ports ouverts. Par conséquent, elles sont facilement détectées et enregistrées par les IDS. Ces techniques tentent d'éviter les dispositifs de filtrage en utilisant d'autres que SYN pour apparaître comme du trafic légitime. Toutes ces techniques ont recours au mappage inverse pour déterminer les ports ouverts. Elles sont SYN- ACK Scan, IDLE Scan, Fin, Xmas Tree et Null Scans, ACK Scan, Windows Scan et TCP Fragmentation Scan[24].
- Sweep Scan : Le scan par balayage, qui ne visent pas à identifier les ports actifs mais plutôt à identifier les hôtes actifs. Ils sont caractérisés comme effectuant des balayages, puisque leur but est d'identifier le statut d'autant d'hôtes que possible au lieu de se concentrer sur un hôte individuel. Ils fonctionnent en générant toute requête susceptible de provoquer une réponse des stations distantes. Ils peuvent être définis comme des facilitateurs de cyber-scanning car ils repèrent les hôtes actifs juste avant que les techniques de balayage réelles des hôtes actifs prennent places. parmis eux (ICMP Echo Request Scan, ICMP Timestamp v Address Mask Scans et TCP SYN Scan. Scans et TCP SYN Scan)[24].
- Miscellaneous Scans : aperçu de la cybernétique en mettant en évidence les scans qui traitent divers protocoles. Il s'agit notamment des scans FTP, UDP, protocole IP et RPC[25].

Après la récolte d'informations, le pirate peut lancer une attaque. Il existe plusieurs techniques d'attaques connues telles que: IP Spoofing, ARP Spoofing, DNS Spoofing, TCP Session Hijacking, Man in the middle, le déni de service distribué[22].

- IP Spoofing : L'usurpation d'identité IP est l'un des domaines les plus étudiés depuis 2005. Il existe plusieurs articles de recherche et systèmes logiciels commerciaux disponibles pour protéger un réseau contre les attaques par usurpation d'adresse IP. Cependant, jusqu'à aujourd'hui, aucun réseau n'est à l'abri des attaques par usurpation d'adresse IP. L'usurpation d'adresse IP n'est pas seulement exclusive au réseau réseau IPv4 mais également applicable au réseau IPv6 en l'absence d'IPSec.

Même s'il existe Plusieurs articles de recherche et produits commerciaux sont disponibles dans la littérature pour contrer cette attaque. articles que nous avons examinés partent du principe que l'attaque provient d'un réseau externe. Aucun des travaux de recherche que nous connaissons dans la littérature ne traitent des attaques par usurpation d'adresse IP qui proviennent de l'intérieur du le réseau, soit par un hôte interne malveillant, soit par des nœuds internes qui se sont associés à des hôtes externes, ou des nœuds internes compromis[26].

- ARP Spoofing : L'usurpation du protocole de résolution d'adresses (ARP) est une menace de sécurité représentative à l'ère d'Ethernet. Dans la procédure d'usurpation d'adresse ARP, un attaquant exploite les failles du protocole ARP pour falsifier l'adresse de l'hôte victime, ce qui lui permet d'obtenir les messages qui lui appartiennent. Bien que simple, ce type d'attaque existe depuis l'avènement d'Ethernet, et il est encore difficile de trouver un mécanisme de défense efficace[27].
- DNS Spoofing : L'usurpation de nom de domaine de serveur, également appelée empoisonnement du cache DNS, est une attaque dans laquelle des enregistrements DNS modifiés sont utilisés pour rediriger le trafic en ligne vers une ressource frauduleuse qui peut ressembler à la destination prévue. Le trafic est ainsi détourné vers l'ordinateur de l'attaquant[28].
- TCP Session Hijacking : Le détournement de session se produit lorsque la session valide d'un utilisateur entre deux ordinateurs est prise en charge par un attaquant[29].
- Man in the middle : L'homme du milieu est un type d'attaque par écoute où les attaquants écoutent secrètement la conversation entre deux utilisateurs légitimes. En cas de besoin, l'attaquant se fait passer pour un utilisateur légitime et pirate les données ou les informations pour les manipuler. Normalement, lors d'une attaque de type "Man in the middle", l'intrus cible en temps réel les transactions, conversations ou transferts d'informations[30].
- Deni de service distribué : L'attaque par déni de service distribué (DDoS) est un essai d'intimidation inondé sur l'Internet. Lors d'une attaque DDoS, la bande passante du réseau présente des victimes (les machines et les ressources) qui sont épuisées pour l'envoi de nombreux paquets vers un serveur ciblé[31].

1.7.2 Les attaques applicatives

Ces attaques sont souvent dues à une faille d'un logiciel ou d'une configuration. La première est essentiellement liée à des injections SQL ou à des buffers overflows.

Ces derniers permettent l'utilisation de shellcode qui donne la possibilité d'exécuter un code à distance. Les injections SQL sont des introductions de code SQL malveillant dans des requêtes de base de données permettant d'obtenir des informations privées. Enfin,

les problèmes dus à une mauvaise configuration sont très répandus. En effet, beaucoup d'administrateurs réseaux préfèrent laisser la configuration par défaut que de la modifier en risquant de mal l'établir.

1.8 Conclusion

Les attaques utilisées par les pirates sont très variées. Certaines utilisent des failles réseau et d'autres des failles de programmation. Nous pouvons donc facilement comprendre que la détection d'intrusions doit se faire à plusieurs niveaux du system. Dans ce chapitre, nous avons décrit les systèmes de détection d'intrusions (Intrusion Detection System-IDS). Nous avons également étudié les différents types d'IDS selon différents critères de classification tels que les méthodes de détection. Afin de formaliser notre système de détection d'intrusion, nous utilisons une ontologie de domaine. Pour cela, nous introduisons les différentes étapes de conception de notre ontologie et ses différents concepts dans le prochain chapitre.

2. Chapitre II Notion de base sur l'ontologie

2.1 Introduction

Le terme «ontologie» est défini en premier temps par les philosophes pour indiquer la métaphysique et l'étude de l'existant. Au fil du temps, le terme a évolué pour se retrouver dans le domaine médical où l'ontologie désigne le traitement de la genèse des maladies. Finalement, la notion d'ontologie s'est étendue au domaine informatique.

Le développement des ontologies représente un espace multidisciplinaire comportant l'apprentissage machine, la représentation des connaissances, l'exploration des données et le traitement du langage naturel. Cela a permet la naissance de plusieurs travaux de recherche dans le domaine de l'ingénierie des ontologies, surtout sur le plan de conception et de développement des ontologies ainsi que la réutilisation de celles-ci. Cependant l'utilisation des ontologies s'étend à plusieurs domaines, et surtout en intelligence artificielle puisqu'elle permet essentiellement de représenter les connaissances et de raisonner sur celles-ci.

2.2 Définition de l'ontologie

2.2.1 Définition dans la littérature

La littérature est pleine de définitions différentes du terme ontologie. Chaque communauté adopte sa propre interprétation selon l'usage qui en est fait et le but visé.

Définition 2.1. Une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire [32].

Définition 2.2. Une ontologie est une spécification explicite d'une conceptualisation, elle est modélisée en mettant en place cinq types d'éléments [33], à savoir:

- Les concepts sont représentés sous forme taxonomique en utilisant des super et/ou sous-concepts ainsi que des concepts composés et/ou élémentaires.
- Les relations sont similaires aux associations du diagramme de classe UML. En effet, ces relations correspondent aux liens entre les concepts.
- Les instances désignent les objets.

- Les axiomes servent à accentuer la vérité en exploitant la logique mathématique.
- Les fonctions correspondent à une relation spécifique entre les différents concepts.

En plus, voici quelques critères de base pour réaliser une ontologie, sachant que ces critères sont variables selon l'objectif de la conception de l'ontologie[34][35]:

- La clarté: les termes utilisés au sein de 1'ontologie doivent être clairs, compréhensibles et objectifs;
- La cohérence: une ontologie doit déduire des faits et la cohérence doit exister sur le plan de la logique entre les concepts et les relations;
- L'extensibilité: une ontologie doit supporter certains ajouts et mises à jour prévus;
- Le biais d'encodage minimal: la conceptualisation et le développement d'une ontologie doivent s'effectuer indépendamment des langages de codage et aussi indépendamment des symboles particuliers. Cela tient essentiellement à la diversité des sources de partage des ontologies.
- L'engagement ontologique minimal: une ontologie doit mettre l'accent sur le minimum d'engagement satisfaisant pour couvrir un domaine prédéfini. L'engagement ontologique s'articule sur la cohérence des termes.

Définition 2.3. Une ontologie est une spécification formelle et explicite d'une conceptualisation partagée. L'ontologie se traduit par un langage compréhensible par la machine [36].

Définition 2.4. Une ontologie est une compréhension partagée d'un domaine d'intérêt, et d'une autre manière est une description formelle d'entités et leurs propriétés, relations, contraintes, et comportement [37].

Définition 2.5. En intelligence artificiel (IA), une ontologie représente un artefact d'ingénierie, constitué par un vocabulaire spécifique utilisé pour décrire une certaine réalité, accompagné d'un ensemble d'hypothèses implicites concernant la signification des mots de ce vocabulaire.

2.3 Les types d'ontologies

2.3.1 Top-Ontologie

C'est le niveau le plus élevé. Il structure les connaissances de haut niveau avec des catégories dont l'organisation dépend des réflexions philosophiques. Elle contient des objets et non des structures, ne s'instancie pas, et se spécialise, c'est à dire, elle spécifie les objets les plus généraux du domaine, les autres objets en seront des spécialisations et non des instances. Les ontologies les plus connues de ce type sont **SUMO, DOLCE, etc**.

Exemple 2.1 (SUMO (Suggested Upper Merged Ontology))**.** *Tous les termes de SUMO descendent des Entitées. Une branche, Entités abstraites, comprend Relations, Prédicats, Sous-classes et Instances. Les entités abstraites forment l'ontologie structurelle, qui nous permet de construire des axiomes (et des règles, comme décrit ci-dessous) dans le SUO-KIF.*[1] *Un exemple d'un axiome typique est : (instance thisGasEngine GasolineEngine). Cela se lit comme suit : "thisGasEngine est une instance de la classe GasolineEngine". "instance" est un prédicat binaire qui relie l'instance à la classe*[2] *[38].*

Définition 2.6. L'ontologie DOLCE (Descriptive Ontology for Linguistique and Cognitive Engineering), élaborer par l'équipe de Nicola Guarino (LOA, Trento, Italie), constitue un des résultats du projet européen WonderWeb Fundation Ontologies Library. DOLCE est une ontologie de haut niveau (niveau supérieure) dont la vocation est d'être utilisée pour concevoir des ontologies de domaine. Sa structure repose sur la distinction philosophique entre entités perdurantes et endurantes. [39]

2.3.2 Core-Ontologie

Ce type d'ontologie fournit des concepts structurant du domaine et les relations entre ces concepts. Par exemple, dans le domaine médical, il y'a des concepts de diagnostic, structure anatomique, et des relations comme celles liées à la localisation d'une pathologie sur une structure anatomique.

Cette Ontologies générée par un type thésaurus qui est Spécifié d'un vocabulaire de référence (physique, math, etc.) et d'une Hiérarchies de termes, relations sémantiques entre les termes(Synonymes, composition, etc.). Un exemple de ce type d'ontologie

Définition 2.7. Mikrokosmos est utilisé dans les traductions automatiques basées sur la connaissance. Il contient des définitions de concepts qui correspondent à des classes d'objets et d'événements dans le monde. Le modèle du monde est organisé en objets, les événements et les propriétés sont placés dans une hiérarchie complexe[40].

2.3.3 Ontologie du domaine

L'ontologie du domaine fournit les concepts du domaine tels qu'ils sont manipulés par des professionnels du domaine, tel que l'ontologie descriptive qui est sémantiquement riche.

[1] Le language dans le quel est écris SUMO

[2] Par convention, les instances sont généralement identifiées par des lettres minuscules. Il faut parfois être prudent pour interpréter les déclarations. Par exemple, "attribut" est un prédicat alors que "Attribut" est une entité abstraite.

Elle définie des concepts et des relations entre ces concepts pour construire des ontologies spécialisées à partir d'autres ontologies plus générales du type **core-ontologies** décrivant un domaine donné par exemple l'ontologie **TOVE** développée dans le cadre du projet d'entreprise virtuelle de Toronto, qui vise à créer une infrastructure d'entreprise sous la forme d'un modèle d'entreprise qui aura la capacité de déduire des réponses aux questions sur les tâches dans un environnement industriel[41].

Les type d'ontologies sont schématisés dans la figure 3.

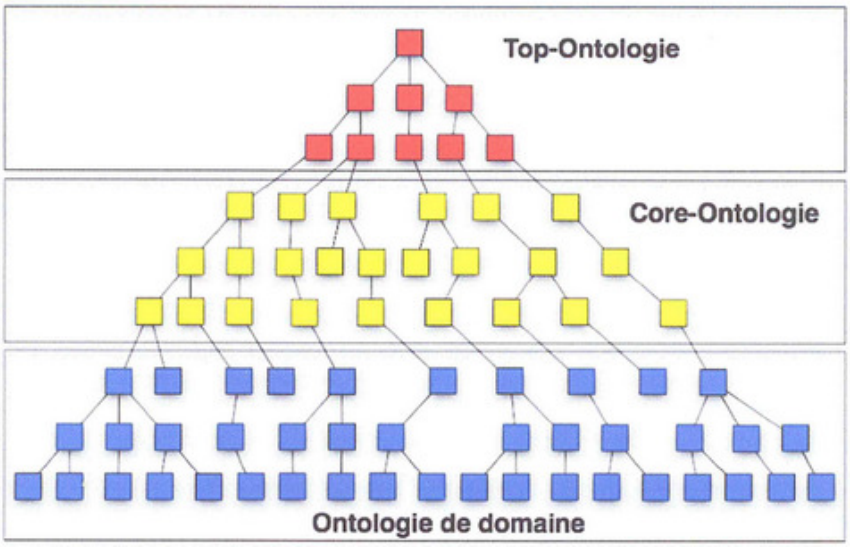


Figure 3. *Types d'ontologies[42]*

2.4 Taxonomie ou Taxinomie des ontologies

Taxinomies : du grec « taxis » : arrangement, ordre (taxon : classe) science des lois de la classification

- La propriété essentielle d'une taxonomie est qu'elle possède une structure hiérarchique.
- Une taxonomie classe selon des propriétés internes aux données, une classification peut être faite selon des critères externes.
- Les taxonomies ont tendance à se mélanger avec les ontologies simples. Utiliser des termes plus spécifiques que le simple terme "ontologie". (comme "ontologie avec hiérarchie d'héritage") aide à donner plus de clarté.
- Lorsqu'un grand nombre de propriétés et de relations sont ajoutées à une structure

hiérarchique, le terme "ontologie" est plus approprié que celui de "taxonomie".

- Une classification vous indique dans quelle case se trouve votre donnée, une ontologie vous indique ce que sont vos données.

2.5 Langages de spécification d'ontologies

L'ontologie peut être éditée avec l'éditeur d'ontologie Protégé et implémentée avec les techniques OWL DL. Avant l'édition de l'ontologie avec Protégé, on choisi le langage de spécification. Par exemple nous pouvons choisir le langage OWL DL, d'une part, l' OWL DL permet d'exprimer des cardinalités multiples et d'autre part, les autres langages sont insuffisants ou plus complexe. RDFS ne permet pas d'exprimer des contraintes de cardinalité. De même, OWL Lite ne permet d'exprimer que des contraintes simples de cardinalité 0 ou 1. Au contraire, OWL Full offre un plus haut niveau d'expressivité, tandis qu' OWL DL offre un niveau d'expressivité assez suffisant. En plus, le codage d'une ontologie sous format OWL présente l'avantage de rendre cette ontologie réutilisable, grâce à l'utilisation des propriétés d'équivalence, de disjonction entre les concepts et les relations. Cette ontologie est vérifiée, finalement, à l'aide du raisonneur.

2.5.1 RDF

RDF [43] (Resource Description Framework) est un langage pour la représentation de métadonnées des ressources. Le modèle RDF permet cette représentation par des assertions sous la forme d'un triplet (ressource, propriété, valeur), ou encore (sujet, prédicat, objet).

- Ressources : les ressources sont tous les objets décrits par RDF. Généralement, ces ressources peuvent être aussi bien des pages Web que tout objet ou personne du monde réel. Les ressources sont alors identifiées par leur URI (Uniforin Resource Identifier).
- Propriétés : une propriété est un attribut, un aspect, une caractéristique qui s'applique à une ressource. Il peut également s'agir d'une mise en relation avec une autre ressource.
- Valeurs les valeurs en question sont les valeurs particulières que prennent les propriétés. La valeur pouvant être une autre ressource ou bien un littéral.

Exemple 2.2. *Sami a 23 ans et habite a constantine*

```
<rdf:RDF>
<rdf:Description about='Sami'>
<rdf:Property about='ville'>
```

```
Constantine
</rdf:Property>
<rdf:Property about="age’>
23
</rdf:Property>
</rdf:Description>
</rdf:RDF>
```

2.5.2 RDF-Schema(RDFS)

RDFS [44] est un langage permettant de définir des schémas de méta-données. Il définit le sens, les caractéristiques et les relations d’un ensemble de propriétés. La principale nouvelle notion est la distinction entre une classe (concept d’une ontologie) et une instance (individu d’une ontologie). Quelques notions définies sont : (rdfs : Class), (rdfs : subClassOf), (rdfs : domain), et (rdfs : range).

L’exemple de la Figure 4 définit le concept de personne, une taxinomie de concepts, et l’instance Sami.

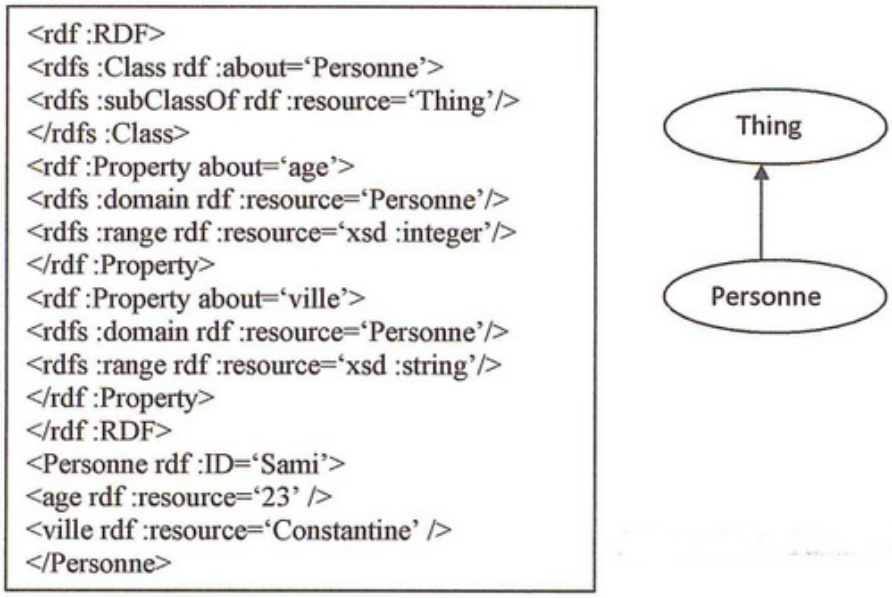


Figure 4. *Exemple de RDFS*

2.5.3 OWL (Web Ontology Language)

OWL [45] est un langage fondé sur la syntaxe RDF/XML et héritier des travaux de DAML+OIL. OWL (Web Ontology Language) introduit l'aspect sémantique qui manque RDF, et offre, par ses primitives plus riches, aux machines une capacité d'interprétation plus grande que celle de RDF et RDFS. OWL se compose de trois sous-langages OWL Lite, OWL DL et OWL Full, qui sont es algorithmes décidables existent pour la totalité de OWL-Lite. Quoique les problèmes d'inférence d'OWL-DL puissent être résolus en temps exponentiel de façon générale, le comportement est souvent satisfaisant. Il n'existe aucun algorithme d'inférence décidable pour OWL-Full. La deuxième version d'OWL étend notamment OWL-DL pour permettre certaines formes simples de métamodélisation, en l'occurrence de créer des concepts de concepts. Ils offrent des capacités d'expression croissantes, chacun est une extension par rapport à son prédécesseur plus simple.

2.6 Base de Données à Base Ontologique

Dans de nombreux domaines, les ontologies sont utilisées pour expliciter la sémantique des données manipulées dans une application. La forte volumétrie des données décrites par des ontologies a entraîné le problème de passage à l'échelle. En conséquence, un nouveau type de base de données a été créé, appelé Bases de Données à Base Ontologique. On appelle base de données à base ontologique une source de données qui contient des ontologies, un ensemble de données et des liens entre ces données et les éléments ontologiques qui en définissent le sens. Les données (instances) contenues dans une Base de données a base ontologique sont appelées des données (instances) à base ontologique. Une Base de données a base ontologique possède deux caractéristiques :

- Les ontologies et les données sont toutes deux, représentées dans une unique base de données et peuvent faire l'objet des mêmes traitements (insertion, mise à jour, interrogation, etc.)
- Toute donnée est associée à un élément ontologique qui en définit le sens et vice versa.

En examinant ces Base de données a base ontologique nous avons identifié une grande diversité concernant les éléments suivants :

- le modèle d'ontologies supporté.
- le schéma de base de données (modèle de stockage) utilisé pour stocker des ontologies .

- le schéma de base de données utilisé pour représenter les données à base ontologique (instances).
- les architectures utilisées pour représenter l'ensemble des informations.

Exemples :

- **BabelNet**, une ontologie réseau sémantique multilingue très grande lexicalisée dans de nombreux langages
- **Base**, est une ontologie d'entreprise. Elle est développée par le Monde Sémantique et permet notamment de créer un graphique de connaissances et un système de coûts
- **Dublin Core**, une ontologie simple pour les documents et la publication
- **FOAF (Friend of a Friend)**, une ontologie pour décrire les personnes, leurs activités, et les relations avec d'autres personnes ou objets
- **CIDOC Conceptual Reference Model** une ontologie pour le patrimoine

2.7 Les ontologies dans la sécurité informatique

Dans ces dernières années, les ontologies dans la sécurité informatique sont largement utilisées vue le besoin d'un partage sémantique de connaissance. Nous allons présenter trois utilisations des ontologies dans le domaine de sécurité informatique.

- RBAC (Role-Based Access Control)
- KAoS (Knowledge Acquisition in autOmated Specification)
- Rei (prononcé en anglais « ray », un mot japonais qui signifie "universel")

2.7.1 Contrôle d'accès à base de rôles (RBAC)

Un rôle représente une fonction spécifique au sein d'une organisation et peut être considéré comme un ensemble d'actions ou de responsabilités associées à cette fonction. Dans un modèle RBAC, toutes les autorisations de concession portent sur les rôles, plutôt que d'être accordées directement aux utilisateurs. Les utilisateurs deviennent alors membres des rôles et acquièrent ainsi les autorisations de ces derniers. L'accès des utilisateurs aux ressources est contrôlé par rôles ; chaque utilisateur est autorisé à jouer certains rôles et, en fonction de son propre rôle, il peut accéder aux ressources et les exploiter en conséquence. Comme un rôle organise un ensemble d'autorisations connexes, il peut simplifier la gestion des autorisations. Lorsqu'un utilisateur a besoin d'un certain type d'autorité pour réaliser une activité, il lui suffit de se voir accorder l'autorité d'un rôle approprié, plutôt que de lui attribuer directement les autorisations spécifiques. En outre, lorsqu'il change sa fonction

au sein de l'organisation, il doit révoquer la fonction d'autorisation du rôle. Les opérations compliquées de révocation d'autorisation en cascade ne sont plus nécessaires. RBAC garantit que seuls les utilisateurs autorisés ont accès à certaines données ou ressources. La hiérarchie des rôles dans RBAC est une façon naturelle d'organiser les rôles afin de refléter les lignes d'autorité et de responsabilité de l'organisation. Par convention, les rôles juniors apparaissent au bas des diagrammes de rôles hiérarchiques et les rôles supérieurs en haut. Les diagrammes hiérarchiques sont des ordres partiels, ils sont donc réflexifs, transitifs et antisymétriques, Plusieurs modèles RBAC sont fournis lorsqu'ils intègrent des contraintes, des sessions et d'autres informations dans le modèle de base[46][47][48].

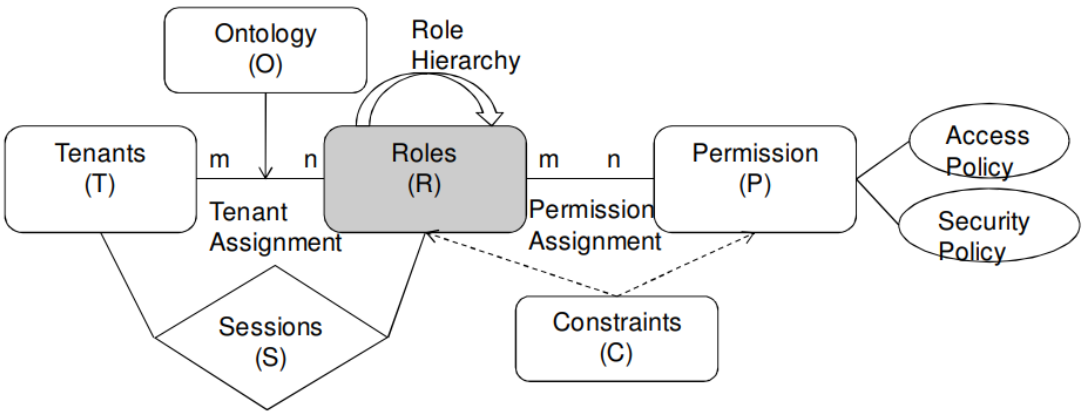


Figure 5. *Ontology pour RBAC*

2.7.2 KAoS

KAoS est une approche de gestion d'agent basée sur differentes politiques, comprenant les problèmes typiques de sécurité (les politiques d'autorisation, le cryptage, politique d'accès et de contrôle de ressources) et des notions supplémentaires (politiques de conversation d'agent, représentations et mécanismes d'application pour des politiques de mobilité, des politiques d'enregistrement du domaine, des politiques d'engagements, etc.)[49].

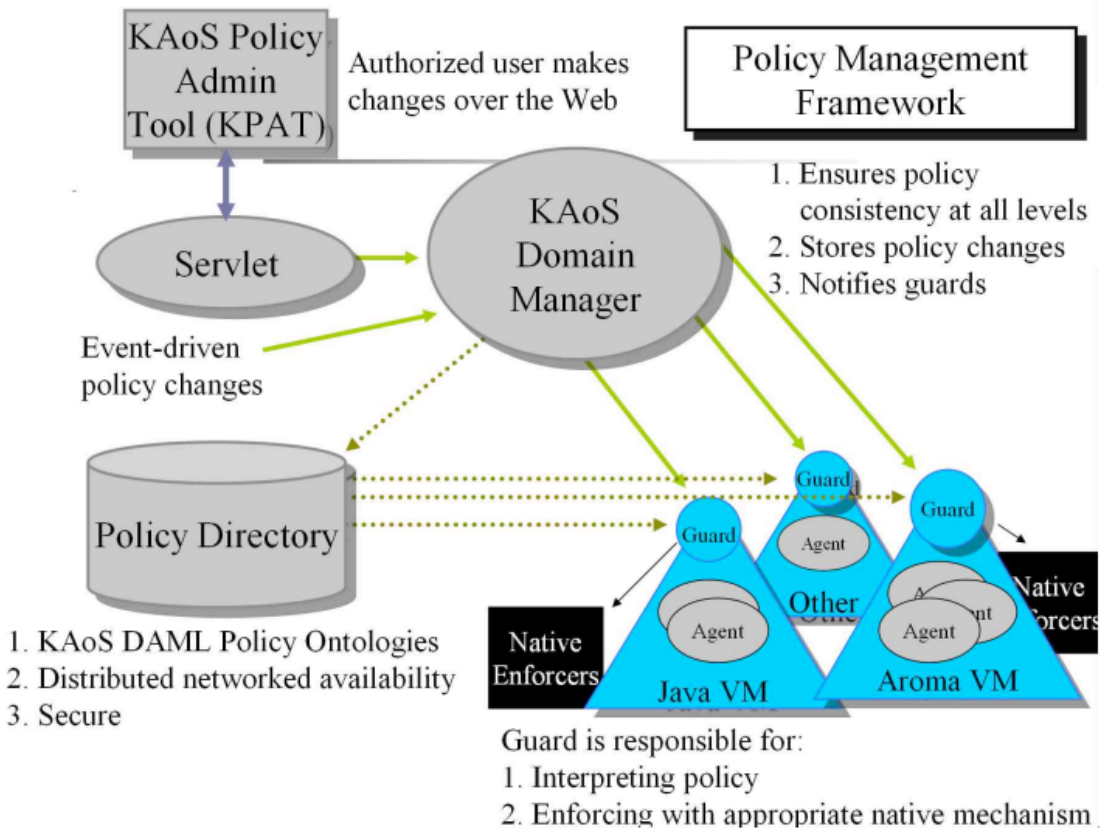


Figure 6. *Architecture du service de politiques KAoS*

2.7.3 Rei

Le langage de politique basé sur des concepts déontiques Rei permet aux utilisateurs d'exprimer et de représenter les concepts de droits, d'interdictions, d'obligations et de dispenses. Ces concepts correspondent, respectivement, aux conditions d'autorisation positive et négative, et d'obligation positive et négative dans KAoS. En outre, Rei permet aux utilisateurs de spécifier des politiques qui sont définies comme des règles associant une entité d'un domaine géré à son ensemble de droits, d'interdictions et d'obligations. Rei s'appuie sur une ontologie indépendante de l'application pour représenter les concepts de droits, interdictions, obligations, dispenses et règles de politique. Cela permet à différents éléments d'un environnement pervasif de comprendre et d'interpréter les politiques de Rei de la manière correcte. L'ontologie de base comprend également la description des actions. Une action générale est décrite par son identificateur d'action unique, les objets cibles sur lesquels l'action peut être exécutée, un ensemble de conditions préalables qui doivent être vrais pour que l'action puisse être exécutée, et les effets qui résultent de l'action lorsqu'elle est exécutée[50][51].

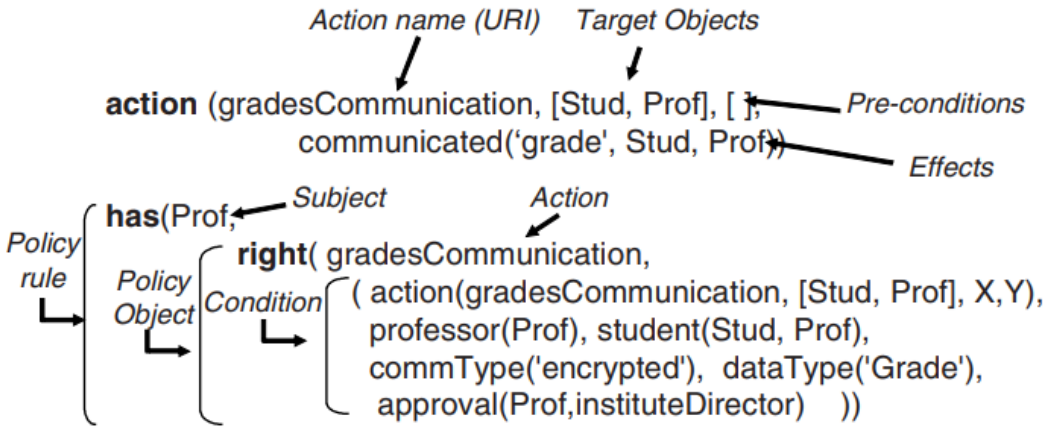


Figure 7. *Spécification des politiques dans Rei*

Le tableau 1 Explique les différentes caractéristiques de KAoS et Rei.

	KAoS	**Rei**
Base ontologique	Oui	Oui
Langage de description	DAML/OWL	Prolog-like syntax + RDF-S
outils pour la spécification des politiques	KPAT - Éditeur graphique pour la gestion des ontologies et des politiques	Non
support de raisonnement	Java Theorem Prover	Prolog engine
Mécanismes d'application	Nécessité d'écrire le code des agents d'exécution appropriés et de les insérer dans les entités à contrôler	L'exécution de l'action est en dehors du moteur Rei
Flexibilité	L'ontologie peut être étendue avec des descriptions d'entités locales dépendant du domaine	L'ontologie peut être étendue avec des descriptions d'entités locales dépendant du domaine

Table 1. Caractéristiques de KAoS et Rei.

2.8 Conclusion

Les ontologies apparaissent désormais comme une clé pour la manipulation automatique de l'information au niveau sémantique. Au fur et à mesure des recherches, des idées se dégagent autour du contenu des ontologies, des méthodes à utiliser pour les construire et des modèles et langages servant à leur représentation.

Dans ce chapitre, nous avons essayé d'éclaircir la notion d'ontologie en présentant plusieurs définitions. Nous avons cités quelques ontologies utilisées dans le domaine de sécurité.

3. Chapitre III La détection d'intrusions et l'ontologie

3.1 Introduction

Dans notre contexte de travail, nous nous concentrons sur les approches de cybersécurité qui prennent en compte la phase d'identification, de détection, de sauvegarde des signatures de l'intrusion afin de proposer un modèle ontologique qui prends en considération l'aspect sémantique des données dans une politique de sécurité informatique. Cette dernière nécessite l'exploitation des données ontologiques en termes de signatures, vulnérabilités, contexte géographique, temporel, politique et/ou économique des attaques, etc. dans le raisonnement ontologique à base de règles d'inférence. Pour ce faire, nous avons effectué un état de l'art concernant les ontologies de domaine existantes dans notre contexte de recherche des différentes phases susmentionnées. En effet, plusieurs travaux de recherches à base d'ontologies on été réalisés dans la littérature à savoir, sur la gestion de vulnérabilité dans la phase d'identification détaillée dans la section 3.2, sur la détection d'intrusions comme définit dans la section 3.3, et les méthodes de détection à base d'ontologies aussi dans un réseau informatique comme décrit dans la section 3.5.

3.2 Les ontologies de gestion de vulnérabilités

Plusieurs ontologies se concentrant sur les vulnérabilités ont été proposées. Dans le domaine de la sécurité de l'information une ontologie conceptuelle générique a été présenté prenant en considérations les vulnérabilités [5]. L'objectif de l'ontologie est de fournir un modèle et une base de connaissances pour le domaine de la sécurité de l'information. L'ontologie de la sécurité est basée sur le modèle de relation de sécurité décrit dans le manuel du **(NIST-National Institute of Standards and Technology)** en français (INNT-Institut National des Normes et de la Technologie) [4]. Les principaux concepts comprennent l'actif, la menace, la vulnérabilité et le contrôle. Les auteurs ont toutefois étendu l'ontologie à des concepts supplémentaires en se référant à plusieurs sources [52] [53] [54] [55]. L'ontologie a été évaluée par un groupe d'experts à l'aide de questions de compétences formelles et informelles. Bien que certaines questions se soient concentrées sur les relations entre les vulnérabilités et les contrôles associés, l'ontologie ne se concentre pas sur la représentation complète de la vulnérabilité. Une ontologie pour la gestion de la vulnérabilité **(OVM-Ontology Vulnerability Management)** a été présentée dans [56] [57]. Informée par l'approche d'ingénierie des connaissances de la logique de description [58], l'ontologie conceptualise le domaine de la vulnérabilité basé sur le **(CVE-Common**

Vulnerabilities and Exposures) ou CVE est un dictionnaire des informations publiques relatives aux vulnérabilités de sécurité et les normes connexes. L'OVM capture des concepts importants tels que la vulnérabilité, l'attaque et la contre-mesure, ainsi que les relations entre les concepts pour caractériser la vulnérabilité des logiciels. L'ontologie est alimentée par les données de vulnérabilité du **(NVD-National Vulnerability Database)**. La logique formelle codée dans OVM permet d'évaluer le niveau de sécurité des produits logiciels ainsi que d'identifier des vulnérabilités similaires. Bien que l'OVM soit l'une des premières tentatives de formalisation du domaine de la gestion des vulnérabilités. l'ontologie est limitée au NVD et ne prend pas en compte les concepts de vulnérabilité provenant d'autres sources importantes telles qu'un utilisateur qui a accès au système [5]. De plus, les concepts de l'ontologie et la capacité de raisonnement ne sont pas évalués pour une représentation adéquate du domaine. Une ontologie de vulnérabilité pour analyser l'effet des vulnérabilités sur un système informatique a été proposée. Cette ontologie est conçue pour aider les analystes de sécurité à obtenir des exigences de sécurité liées aux vulnérabilités. Cependant, l'ontologie est un méta-modèle abstrait et la correction et l'utilité de l'ontologie ne sont pas évaluées[59]. Des systèmes basés sur l'ontologie pour prédire et classifier les attaques d'applications Web ont également été proposés [60]. Les auteurs utilisent une approche basée sur l'ontologie pour prédire les attaques en utilisant le contexte des vulnérabilités, des menaces et des conséquences. Cette étude propose trois modèles ontologiques à savoir, un modèle de menace, un modèle de vulnérabilité et un autre modèle d'attaque. Les modèles sont liés pour dériver des inférences et prédire les attaques.

3.3 Les ontologies de détection d'intrusions

Une ontologie pour la détection d'intrusions qui capture la sémantique pour les signatures des attaques, mais sans se limiter a l'état du système tels que l'état de la connexion, l'utilisation de l'unité centrale et de la mémoire, l'adresse IP, le port de la machine, le protocole et les journaux du routeur et du pare-feu [61]. Les représentations des vulnérabilités dans cette ontologie sont alignées sur les codes CVE courants. L'ontologie peut être utilisée pour la détection des intrusions par signature en utilisant une correspondance de chaînes de caractères, et la détection coopérative multi-capteurs.

3.4 Les ontologies pour sécurisé les opérations et les processus

Une ontologie formellement fondée pour les opérations sécurisées [62]. Elle peut être utilisée, entre autres pour décrire la tâche de récupération de fichiers en toute sécurité et pour détecter les intrusions. Elle présente des concepts tels que cyber-opération, cyber-

opérateur, plan de mission, cyber-exploitation, charge utile, etc. et les relations entre ces concepts. Une ontologie pour les exigences de cybersécurité des processus[63]. Elle définit les concepts à quatre niveaux, dont le plus élevé contient les concepts clés qui sont: la confidentialité, le contrôle d'accès, la disponibilité, l'intégrité, la responsabilité, la détection et la prévention des attaques et des dommages. La vie privée a deux sous-classes directes la confidentialité et le consentement de l'utilisateur. Le contrôle d'accès définit des sous-classes telles que l'authentification, l'identification et l'autorisation. La disponibilité comprend le service, les données, le personnel et les sauvegardes matérielles. L'intégrité couvre le contrôle de l'intégrité des données, du matériel, du personnel et des logiciels. La classe de responsabilité a des sous-classes telles que la piste d'audit, la criminalistique numérique, et la non-répudiation. La détection et la prévention des attaques et des dommages comprend les concepts suivants :

1. évaluation de la vulnérabilité;
2. Honey pot (pot de miel) est un mécanisme permettant de capturer la stratégie d'attaque utilisée par l'attaquant. Il ne permet pas de prévenir ou d'atténuer l'attaque. Le travail du pot de miel est de rester silencieux et de se faire passer pour un véritable système et de piéger l'attaquant. Les pots de miel sont déployés de manière à ce que les attaquants le considèrent comme un système productif et l'attaquent. Il recueille des informations sur l'attaquant par ses mouvements et en donnant les détails requis par l'attaquant. Le seul objectif du Honeypot est de collecter des informations sur l'attaque et les attaquants plutôt que de l'empêcher. Il existe deux types de pots de miel, les pots de miel de recherche et les pots de miel de production;
3. Pare-feu et système de détection et de prévention d'intrusion.

3.5 Les ontologies de réseaux

La représentation formelle des infrastructures de réseaux fournit des schémas de données qui peuvent permettre l'automatisation des tâches via l'apprentissage automatique pour l'analyse des vulnérabilités et des expositions [64]. Ces infrastructures comprennent non seulement des ordinateurs, mais aussi des dispositifs de l'Internet des objets (Internet of Things-IoT), qui peuvent être décrits efficacement à l'aide d'ontologies [65]. Les représentations basées sur les ontologies peuvent également être utilisées pour les réseaux sans fil sans infrastructure, tels que les réseaux mobiles ad hoc **(MANET-Mobile ad hoc networks)** qui est le nom d'un groupe de travail de l'IETF[1], créé en 1998/1999, chargé de standardiser des protocoles de routage basés sur la technologie IP pour les réseaux

[1]L'Internet Engineering Task Force, élabore et promeut des standards Internet, en particulier les standards qui composent la suite de protocoles Internet. L'IETF produit la plupart des nouveaux standards d'Internet.

ad hoc sans fil.[2] qui peuvent être décrits à l'aide de MANET (Distributed Functions Ontology (MDFO)) [66]. Les premières ontologies de gestion de réseau remontent aux débuts du Web sémantique [67]. L'ontologie de gestion de réseaux informatique définit des concepts tels que l'entité de trafic, les événements, la base de données, l'outil de gestion, les statistiques de trafic, l'action, la demande, l'acteur, le routage, l'anomalie et l'entité de réseau [68]. L'ontologie de domaine de réseau a été conçue pour les représentations détaillées de réseau, où les dispositifs de réseau sont des instances de concepts de réseau, qui peuvent être décrits à l'aide de propriétés telles que l'adresse MAC, l'adresse IP, le système d'exploitation, et le numéro du bureau virtuel dans lequel le dispositif est situé [69]. **Network Ontology (NetOnto)** est une ontologie conçue pour décrire les routes échangées **(Border Gateway Protocol-BGP)** [70]. Les axiomes OWL de l'ontologie sont complétés par des politiques BGP écrites sous forme de règles (Semantic Web Rule Language-SWRL), qui déterminent comment les informations de routage sont partagées entre les voisins pour contrôler le flux de trafic sur les réseaux. Ces politiques comprennent également des informations contextuelles telles que les affiliations et les restrictions de route. L'utilisation de NetOnto permet de se concentrer sur les politiques de haut niveau et de reconfigurer les réseaux automatiquement. Une ontologie modulaire qui définit des concepts à trois niveaux de granularité suivants:

1. Premier niveau (Switch, Router, Firewall);
2. Deuxième niveau (Workstation, Server, PublicService);
3. Troisième niveau (SharedWorkstation, PublicServer).

parmi lesquels la deuxième lettre peut être générée automatiquement par un raisonnement sur la première [71].

3.5.1 Les méthode de détection

L'ontologie de détection d'intrusion est centrée sur le système cible ou victime [72]. Comme les auteurs le mentionnent, les systèmes de détection d'intrusion sont souvent décrits du point de vue de l'attaquant et prétendent que le point de vue devrait être celui du système cible, puisque c'est là que la détection réelle a eu lieu. L'ontologie de détection d'intrusions étend ce modèle centré sur la cible et le croise au niveau de l'objet réseau-paquet. La perspective est similaire dans la mesure où elle prend également en compte les facteurs observables et mesurables localement. La différence est qu'il s'agit d'une ontologie de détection d'intrusions basée sur l'attaquant, qui suppose que l'attaquant est connecté localement et qui tente de compromettre des systèmes externes. Une relation de

[2]https://datatracker.ietf.org/doc/html/rfc2501

sous-classe dans OWL, par exemple, se présente comme suit :

```
<owl:Class rdf:ID="ProcessSigner">
<rdfs:subClassOf>
<owl:Class rdf:about="#Signer"/>
</rdfs:subClassOf>
</owl:Class>
```

Tous les attributs de signature, les agents, les classes d'agents, les dépôts de données et les types de protocoles sont définis par l'ontologie de la même manière que dans [72]. Les données sont transférées et stockées selon l'ontologie afin que les opérations logiques puissent avoir lieu.

3.5.1.1 Les signatures OWL

Les signatures sont formatées en utilisant la sémantique de l'ontologie. Un exemple de signature de processus pour un outil d'attaque (apache scalp) :

```
<ProcessSignature rdf:ID=apache-scalp>
<name> apachescalp </name>
<md5> 9d4ad46076f207c90a5438734efb5632 </md5>
<textSegmentSize> 6391 </textSegmentSize>
<dataSegmentSize> 620 </dataSegmentSize>
<arguments> null </arguments>
<numArguments> 2 </numArguments>
<size> 18769 </size>
</ProcessSignature>
```

Remarquez que ces valeurs sont pré-chargées dans la base de connaissances de l'agent lorsque celui-ci est initialisé. Pareillement, une signature de réseau ressemblerait à ceci :

```
<TrafficSignature rdf:ID=pam>
<packetCounter> 16 </packetCounter>
<fragmentation> 0 </fragmentation>
<sourceAddress> -8205 </sourceAddress>
<sourcePort> 0 </sourcePort>
<destinationAddress> -5110 </destinationAddress>
```

```
<destinationPort1> 6023 </destinationPort1>
<destinationPort2> 110 </destinationPort2>
<windowSize> 1 </windowSize>
<deviation> 0 </deviation>
<flags> "0,8,8,8,8,8,8,8,1,1,1,1,1,1,1,1" </flags>
<lengths> "0,0,0,0,0,0,0,0,0,0,6,100,182,0,33,0" </lengths>
<hashes> "0,0,0,0,0,0,0,0,0,0,349,1223,1617,0,1689,1689" </hashes>
<execVariant> 7a </execVariant>
</TrafficSignature>
```

Les fonctions logiques de base exécutées par les agents pour opérer sur ces signatures et les événements observés (concordances, non-concordances, signalement de nouveaux agents, entrée reçue, etc) telles que :

- L'inférence à quatre niveaux : un agent peut exécuter des dérivations directes et transitives jusqu'à quatre niveaux de la hiérarchie;
- L'identification du récepteur : un agent peut identifier les détails taxonomiques du récepteur en utilisant les informations stockées localement dans sa base de connaissances sans demander d'informations supplémentaires;
- Le stockage intelligent des données : lorsqu'un message correspond à un sujet de l'ontologie, l'agent ne stocke pas les informations qu'il connaît déjà sur une entité.

3.5.1.2 Le correspondance de signatures

La méthode de correspondance est un mécanisme de notation. Un score global sigma représente la similitude entre une entrée et une signature déterminée. Ce score est composé d'une combinaison linéaire de sous-scores (sigma i) qui correspondent à une correspondance entre le i-ème attribut de la signature et l'entrée. Chaque attribut contribue différemment à σ. Le poids (ω i) d'un attribut est déterminé par son pouvoir de division, qui indique également sa pertinence en tant que mesure de similarité :

$$\omega i = ln(k) - ln(i)$$

où k est le nombre total d'attributs. Les poids sont combinés dans σ et ensuite normalisés pour générer une sortie dans [0 , 1] :

$$\sigma = \sum_{i=1}^{k} \frac{\sigma_i \omega_i}{\sum_{j=1}^{k} \omega_i}$$

L'intervalle de σi est [0, 1]. En fait, tous les attributs renvoient une valeur booléenne, $\sigma i \in \{0, 1\}$ sauf les attributs de hachage dans les signatures de réseau. ils ne sont pas vérifiés pour une correspondance identique. Comme mentionné précédemment, les sous-protocoles TCP peuvent changer la charge utile d'une séquence et c'est la raison pour laquelle l'attribut est utilisé. Une fonction triangulaire a été utilisée pour évaluer la similarité de ces valeurs de hachage. Le maximum de la fonction correspond avec une déviation nulle (c'est-à-dire une correspondance parfaite). Toute différence entre la signature et le hachage d'entrée sera pénalisée par un score σi plus faible au fur et à mesure que l'écart déviation augmente. Les correspondances dont la déviation est supérieure à celle stockée dans la signature obtiendront une valeur de similarité nulle. La valeur de σi pour un attribut de hachage est calculée comme suit :

$$\sigma_i = 1 - \frac{\mid h_i - h_{input} \mid}{deviation}$$

où (h input) est le hachage MD5 de la charge utile du paquet d'entrée, (h i) représente le i-ème hachage de la signature, et (déviation) est l'attribut de déviation. Par conception, la correspondance exacte de tous les autres attributs est vérifiée. Les adresses IP, les numéros de port, la taille de la fenêtre, etc. sont des caractéristiques pour lesquelles différents degrés de similarité ne fonctionnent pas correctement. Si, par exemple, une signature correspond à un programme d'attaque qui cible toujours un numéro de port déterminé et que l'entrée présente un numéro de port proche du numéro de port de la signature par exemple, le port 22 et le port 23, cette signature ne sera pas prise en compte, cela ne signifie pas que l'entrée ressemble en fait à la signature [73]. Différents protocoles peuvent être exécutés sur des numéros de port différents et toute similitude entre les signatures ne serait qu'une coïncidence. La correspondance des signatures de réseau est effectuée par les cellules **(TMC-Traffic Message Channel)**, où TMC, est une norme européenne qui permet de diffuser des informations de circulation aux automobilistes, généralement via le système de transmission numérique RDS (Radio Data System) utilisant la bande radio FM, alors que les cellules **PMC-Le perceptron multicouche (PMC)** est un réseau composé de couches successives. Une couche est un ensemble de neurones n'ayant pas de connexion entre eux. Une couche d'entrée lit les signaux entrant, un neurone par entrée x_j, une couche en sortie fournit la réponse du système. font correspondre les signatures de processus de manière indépendante. Les attributs donnent un 0 ou un 1 qui activera ou désactivera le poids de l'attribut dans le calcul de σ. Lorsque la recherche atteint γ nœuds, l'entrée est comparée à plusieurs signatures sous un nœud βi déterminé. Le score σ le plus élevé sera choisi pour indiquer la signature à laquelle l'entrée ressemble le plus. First-Fit Renvoie la première correspondance rencontrée lors de la recherche sans tenir compte des correspondances ultérieures, qui peuvent être meilleures et a évitée en comparant l'entrée à un certain nombre de signatures avant de faire une hypothèse.

3.5.2 La fusion de signatures

Les cellules de surveillance traitent les signatures séparément. En effet, elles constituent en elles-mêmes des entités de détection d'intrusion. Les résultats rapportés par une cellule de surveillance de processus (PMC) sont indépendants de ceux obtenus par une cellule de surveillance du trafic (TMC). Si une cellule du réseau tombe en panne est incapable d'effectuer la détection, ou si un outil d'attaque a été exécuté mais que l'interface de réseau est hors service et qu'aucun paquet n'a été généré, une cellule de surveillance des processus peut encore identifier le programme en cours d'exécution afin que le système de sécurité prenne des mesures. De même, en cas de défaillance d'une cellule de processus, les cellules de réseau peuvent fournir un support de détection complet au système en analysant le trafic sortant. Comme le montrent les recherches précédentes, l'intégration de plusieurs sources de preuves est nécessaire pour augmenter la précision d'un système de détection d'intrusion[74][75]. Ceci est particulièrement important lorsque la réponse automatisée est mise en œuvre. Si le système signale comme une activité illicite est en réalité une violation de la politique de sécurité en place, il est possible d'intégrer la réponse automatisée dans le système de sécurité. Lorsque les résultats de la détection ne sont pas entièrement fiables, aucune réponse automatisée efficace ne peut être construite au-dessus du mécanisme de détection des intrusions.

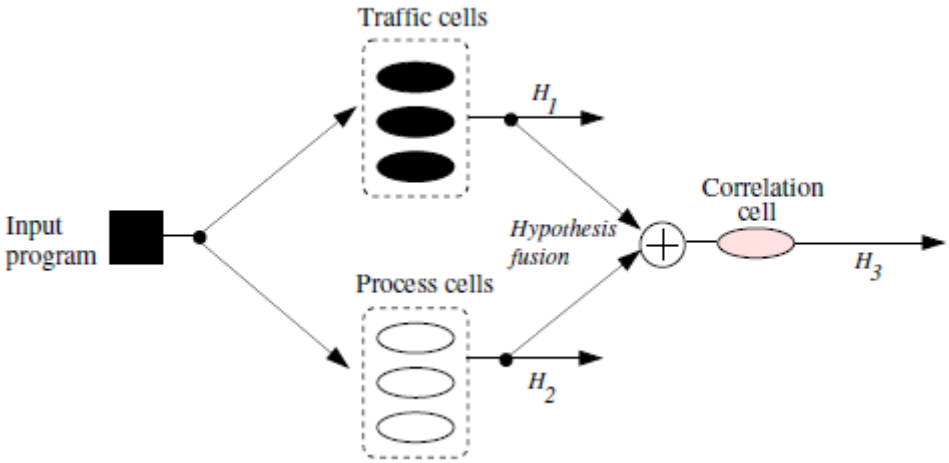


Figure 8. Fusion d'hypothèses H1 et H2 en H3

Pour cette raison, les deux systèmes de détection mis en œuvre sont intégrés par des cellules de corrélation qui déterminent un score unique I qui fait intervenir les deux hypothèses concernant la signature à laquelle correspond l'entrée observée. Comme illustré dans la Figure 8, une hypothèse H_i est la signature que n'importe quelle des cellules de contrôle signale comme la plus probable. L'hypothèse H_1, par exemple, peut correspondre à une signature ayant une des similarités σ, tandis que H_2 peut correspondre à une signature de

processus avec un score de similarité π. Le score de similarité intégré I est calculé par une cellule de corrélation qui reçoit des entrées de deux et seulement deux cellules de contrôle comme suit :

$$(1) IF\{\exists H1 \wedge \nexists H2\} \implies \{H3 = H1 \wedge I = \sigma\}$$

$$(2) IF\{\nexists H1 \wedge \exists H2\} \implies \{H3 = H2 \wedge I = \pi\}$$

$$(3) IF\{\nexists H1 \wedge \nexists H2\} \implies \{H3 = I = NULL\}$$

$$(4) IF\{\exists H1, H2 \mid H1 = H2\} \implies \{H3 = H1 = H2 \wedge I = \frac{\sigma + \pi}{2}\}$$

La cellule de corrélation (CC) met alors en œuvre une réponse automatisée sur la base des informations reçues de deux cellules de surveillance. En fonction des scores de similarité signalés, différentes actions de réponse peuvent être mises en œuvre. Lorsque les cellules rapportent des scores de similarité élevés, le type de réaction à mettre en œuvre peut être plus définitif et repose sur les données fournies par les cellules. Dans le cas où des scores faibles sont rapportés, il sera préférable de procéder en toute sécurité en déclenchant des actions qui n'ont pas de conséquences irréversibles. Trois types de cellules sont capables de réagir :

- Les cellules de surveillance du trafic peuvent réagir aux séquences de paquets qu'elles observent;
- Les cellules de surveillance des processus peuvent réagir à l'exécution d'un processus qu'elles considèrent comme suspect;
- Les cellules de corrélation peuvent agir sur une entrée fusionnée.

Les types de réaction mis en œuvre sont utilisés pour prouver la nécessité d'avoir des niveaux de précision de détection élevés afin de permettre à un système de répondre automatiquement à des événements classés automatiquement.

3.5.2.1 Les types de réponses

- A = envoyer des données sur le trafic à un dépôt de preuves;
- B = déplacer l'exécutable vers le dépôt de preuves;
- C = copier l'historique des commandes dans le dépôt;
- D = envoyer le signal de suspension de processus;
- E = modifier les privilèges d'exécution des fichiers;
- F = envoyer le signal "kill-process";
- G = alerter le système cible.

Le Tableau 2 résume les méthodes de détection d'intrusion à base d'ontologies.

	Ontologie de gestion de vulnérabilités. 3.2	**Ontologie de détection d'intrusions. 3.3**	**Ontologie pour sécurisé les opérations et les processus. 3.4**	**Ontologie réseau. 3.5**
Méthode de détection	L'ontologie est alimentée par les données de vulnérabilité du (NVD-National Vulnerability Database). La logique formelle codée dans l'OVM (Ontology Vulnerability Management) permet d'évaluer le niveau de sécurité des produits logiciels ainsi que d'identifier des vulnérabilités similaires. Mais elle ne prend pas en compte les concepts de vulnérabilité provenant d'autres sources importantes telles qu'un utilisateur qui a accès au système.	Capture la sémantique pour les signatures des attaques, mais sans se limiter a l'état du système. Elle est utilisée pour la détection des intrusions par signature.	Formellement fondée pour les opérations sécurisées. Elle présente des concepts et les relations entre ces concepts. Une ontologie pour les exigences de cyber-sécurité des processus qui définit les concepts à quatre niveaux dont le plus élevé contient les concepts clés qui sont: la confidentialité, le contrôle d'accès, la disponibilité, l'intégrité, la responsabilité, la détection et la prévention des attaques et des dommages.	Étend le modèle centré sur la cible et le croise au niveau de l'objet réseau-paquet. La perspective est similaire dans la mesure où elle prend également en compte les facteurs observables et mesurables localement. Il s'agit d'une ontologie de détection d'intrusions basée sur l'attaquant, qui suppose que l'attaquant est connecté localement et qui tente de compromettre des systèmes externes.

Table 2. Méthodes de détections à base d'ontologies.

3.6 Conclusion

Détecter les intrusions a toujours été une tache pénible et exigeante pour tout ingénieur en cyber-sécurité, car tout comme les systèmes informatique se développent au fil du temps leurs failles se développent aussi, les systèmes de détections d'intrusions suivent des algorithmes dynamiques mais qui ne peuvent pas prendre de décisions et donnent beaucoup de fausses alertes. C'est pour cela qu'on a décider d'intégrer l'ontologie, car grâce aux liens entre les données nous allons pouvoir améliorer grandement les systèmes des détection d'intrusions en diminuant les fausses alertes.

Et pour cela nous nous somme inspirer des différentes ontologies qui existent déjà dans le domaine de la cyber-sécurité pour pouvoir concevoir notre domaine d'application ontologique.

4. Chapitre IV Mise en oeuvre

4.1 Introduction

Les systèmes de détection d'intrusion actuels ont pour principales lacunes un nombre élevé de faux négatifs et de faux positifs. Il arrive que les systèmes de détection d'intrusion se trompent dans leurs détections et que l'inefficacité de la détection soit en partie due à l'insuffisance des données d'audit. Certains systèmes de détection d'intrusion dépendent d'un seul type de sources : les données du réseau ou les données de l'hôte. Pour ces raisons, nous pouvons étendre le champ d'analyse et d'investigation et utiliser plus d'un système pour détecter les intrusions et les activités suspectes dans le réseau. L'utilisation de nouvelles méthodes et techniques dans la phase de détection peut également réduire le taux de faux. L'utilisation d'une **ontologie de domaine** est une nouvelle solution pour la détection des attaques en exploitant les données sémantiques à base de règles de raisonnement ontologique. Pour ce faire, nous avons développé un système qui détecte, identifie et sauvegarde les signatures des cybers attaques dans une base de données ontologique que nous avons conçue. Ce système est un "Serveur Proxy" qui joue le rôle d'intermédiaire en se plaçant entre deux hôtes pour surveiller leurs échanges. La figure ci dessous explique le fonctionnement d'un serveur proxy :

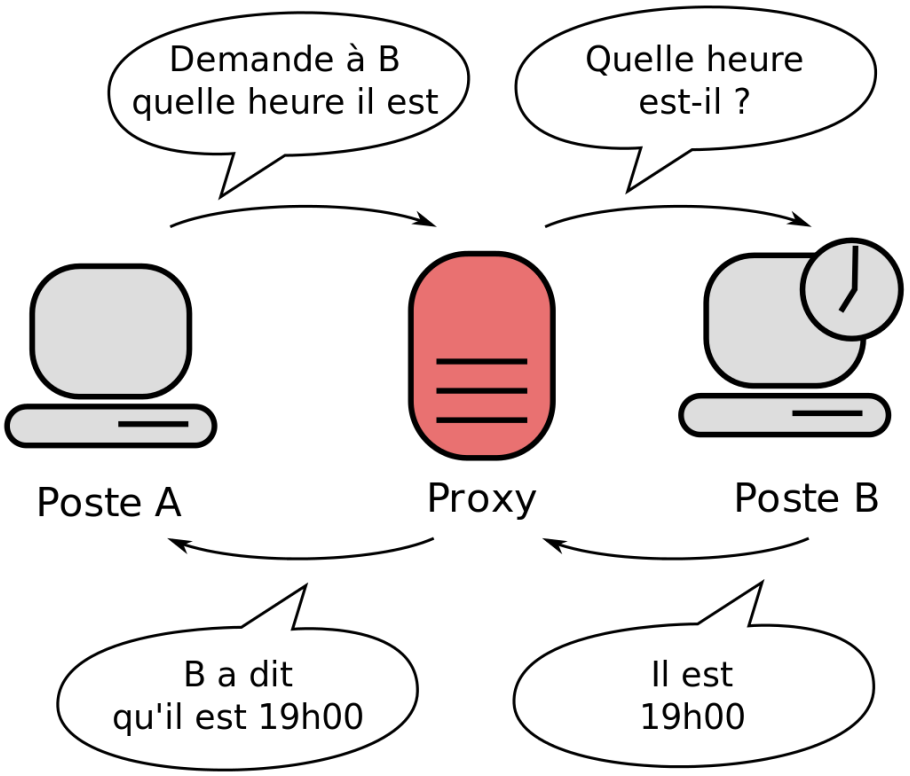


Figure 9. *Principe du serveur Proxy*

Dans ce chapitre nous allons détaillé notre modele ontologique ainsi que les différents outils que nous avons utiliser pour concevoir notre politique de sécurité pour qu'elle puisse détecter, identifier et sauvegarder les signatures des cyber attaques.

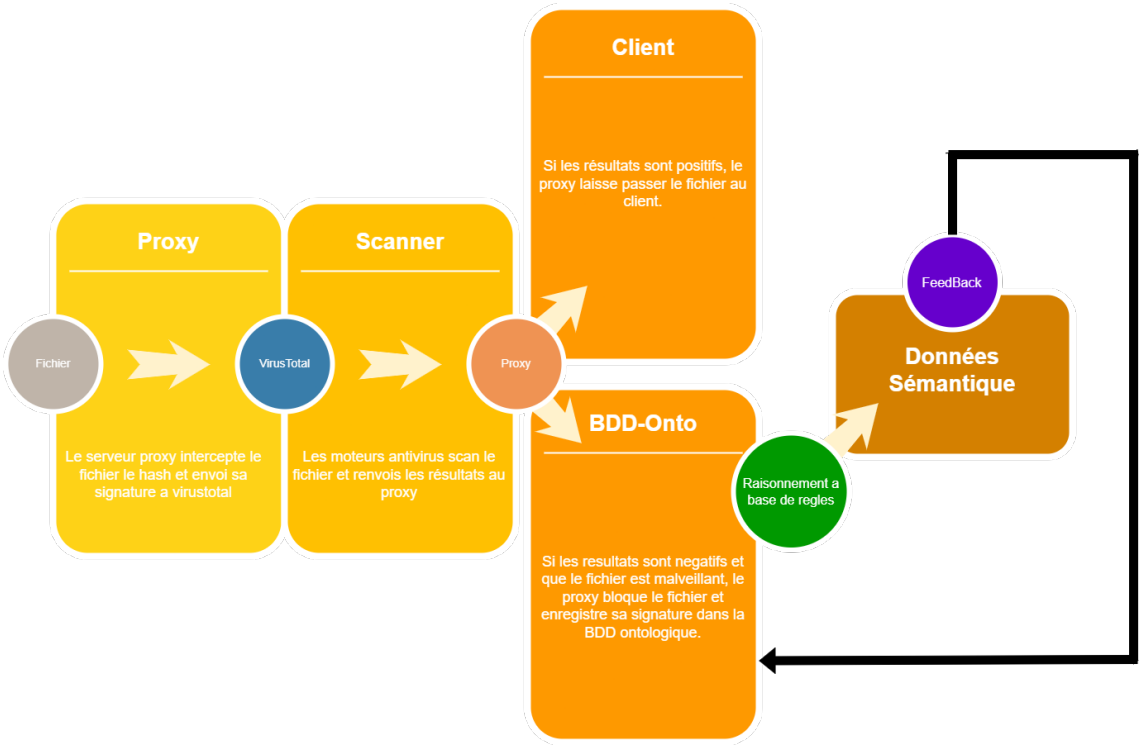


Figure 10. *Architecture de notre système*

1. **Proxy** : Notre serveur proxy intercepte les données, les chiffres en MD5, les scan avec VirusTotal et selon les résultats du scan soit il laisse passer ces données soit il les bloque et enregistre leurs signature dans la base de données ontologique.
2. **VirusTotal** : Service de scan en ligne, avec la multitude de moteurs d'antivirus qu'il utilise pour scan, c'est un des logiciel qu'on puisse utiliser pour scan nos données.
3. **Base do données ontologique** : Notre base de données ontologique est nourrit des meilleurs données grâce a notre proxy, et a son architecture, elle est prête a améliorer les systèmes de détection d'intrusions.

4.2 Plateforme attaquable

Qu'est-ce qu'une plateforme attaquable

C'est un catalogue de vulnérabilités qui sont légalement " cassables, piratables et exploitables ", elle a été conçu dans le but d'entraîner ceux qui veulent en apprendre plus sur la cyber sécurité. Cette plateforme nous l'avons adapter a nos besoins afin de faciliter la compréhension de nos test.

Lien de la plateforme : `https://github.com/Yavuzlar/VulnLab/`

Elle a été programmer avec :

- **PHP** : PHP est similaire aux langages de haut niveau tels que C, Perl, Pascal,

FORTRAN et Java, et facile à apprendre. PHP est un acronyme récursif qui signifie "Hypertext Preprocessor", qui est un langage de script et généralement intégré ou combiné au HTML et possède de nombreuses bibliothèques excellentes qui fournissent un accès rapide et personnalisé aux SGBD. C'est un outil idéal pour développer une logique d'application dans le niveau intermédiaire d'une application à trois niveaux [76].

Pourquoi PHP ?

PHP est un langage de programmation populaire pour les applications web côté serveur et utilisé par Wordpress, Drupal, etc. Il est gratuit et facile a mettre en place dans un environnement de développement et il existe plusieurs projets PHP open-source disponibles dans des dépôts publics qui peuvent nous permettre d'effectuer une évaluation de la sécurité[77].

- **Java Script Object Notation (JSON)** : est un format de données textuelles dérivées de la notation des objets du langage Java Script. Il permet de représenter de l'information structurée comme le permet XML par exemple.

  Pourquoi JSON ?

  Le JSON est peu "verbeux", et peut prendre bien moins de place que XML pour représenter un jeu de données équivalent. Cela assure par exemple une réception plus rapide d'un large jeu de données JSON lorsque l'on fait appel à un web service que si les données étaient au format XML.

4.3 Serveur Proxy

- **Python** : est un langage de programmation puissant et orienté objet [78]. Il expose les raisons de l'accréditation de Python comme le langage de programmation ayant connu la plus forte croissance ces derniers temps, en s'appuyant sur des recherches et des articles publiés dans divers magazines et sites Web populaires. Il présente les caractéristiques et les traits les plus importants du langage Python, les types de systèmes de gestion de bases de données supportés par Python, ses utilisateurs et ses applications. Python est orienté objet car il fonctionne avec des champs et des méthodes) et il a une compatibilité multiplateforme qui peut être programmé avec le même ensemble de fonctionnalités sous Windows ainsi que sous MacOS, Linux, Nix et d'autres systèmes d'exploitation populaires. Le programme peut analyser sa structure et la modifier pendant l'exécution du code. Il est possible d'exécuter directement des instructions et il possède des fonctions de traitement symbolique des données. Le langage est orienté aspect en divisant le programme en modules d'aspect. C'est un paradigme de programmation horizontal où l'on peut ajouter un comportement ou une fonction à plusieurs classes qui ne partagent pas le même héritage vertical orienté objet. Python a une syntaxe minimaliste, mais elle

n'est pas de niveau inférieur et surpasse parfois des environnements programmables plus importants. Le minimalisme permet d'augmenter la vitesse d'écriture des programmes, ainsi que la vitesse de lecture du code. La bibliothèque standard des modules du langage comprend un ensemble toujours plus grand de fonctions différentes, et l'utilisateur peut facilement ajouter des fonctions manquantes ou nouvelles. La raison de la popularité de ce langage est la clarté et la simplicité de sa structure.

Pourquoi Python ?

L'une des plus grandes forces de Python est sa grande bibliothèque standard qui fournit des outils adaptés à de nombreuses tâches. Divers modules permettant de créer des interfaces graphiques, de se connecter à des bases de données, de générer des nombres pseudo-aléatoires, des calculs arithmétiques, des expressions régulières, etc. sont inclus dans la bibliothèque.

4.4 Les logiciel utilisés

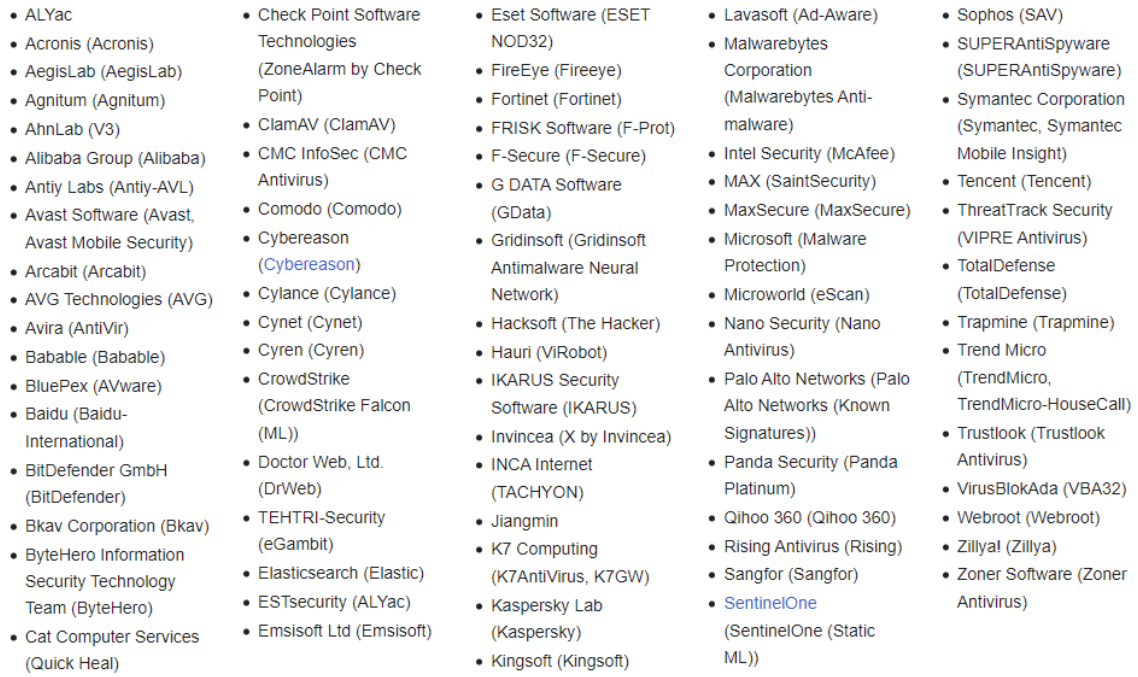
- **Protégé :** Protégé-2000 est un outil de modélisation des connaissances développé à l'université de Stanford. Les ontologies et les bases de connaissances peuvent être éditées de manière interactive dans Protégé et accessibles par une interface utilisateur graphique et une API Java. Protégé peut être étendu avec des composants pluggables pour ajouter de nouvelles fonctionnalités et de nouveaux services. Il existe un nombre croissant de plugins offrant une variété de fonctionnalités supplémentaires, telles que des outils de gestion d'ontologie supplémentaires, un support multimédia, des moteurs d'interrogation et de raisonnement, des méthodes de résolution de problèmes, etc. Protégé met en œuvre un riche ensemble de structures et d'actions de modélisation des connaissances qui permettent la création, la visualisation et la manipulation d'ontologies dans divers formats de représentation. Protégé offre un support pour la construction d'ontologies basées sur des cadres, conformément au protocole Open Knowledge Base Connectivity (OKBC). La version étendue du système basé sur la trame a été introduite en 2003 pour supporter OWL avec un avantage de la version web sémantique. Il existe plusieurs formes telles que RDF(s), OWL et XML Schema dans lesquelles l'ontologie conçu dans Protégé peut être exportée. Voici les différents plugins disponibles dans Protégé[79] :
  - **JSave** : Pour générer des stubs de définition de classe java pour les classes dans Protégé.
  - **Protégé Web Browser** : Permet aux utilisateurs de partager des ontologies conçu dans Protégé sur internet.
  - **WordNet plug-in** : C'est une interface pour la base de connaissances WordNet.
  - **XML Schema** : Transforme une connaissance conçu dans Protégé en XML.

- **UML Plug-in** : Génère un fichier de schéma XML décrivant le modèle de connaissances de protégé et un fichier XML.
- **DataGenie** : Permet de lire et de créer un modèle de connaissances à partir d'un SGBDR[1] en utilisant JDBC[2].
- **Docgen** : Pour créer des rapports décrivant les ontologies.
- **PROMT** : Permet de gérer plusieurs ontologies avec protégé.
- **OWL-S Editor** : Permet de charger, créer, gérer et visualiser les services OWL-S.
- **OntoGraf** : permet de naviguer de manière interactive dans les relations de vos ontologies OWL. Différentes mises en page sont prises en charge pour organiser automatiquement la structure de votre ontologie. Différentes relations sont prises en charge : sous-classe, individu, propriétés d'objet de domaine/plage et équivalence. Les relations et les types de nœuds peuvent être filtrés pour vous aider à créer la vue que vous souhaitez.

■ **VirusTotal** VirusTotal est un service en ligne appartenant à Google qui permet l'analyse de fichiers suspects et facilite la détection rapide des virus, vers, chevaux de Troie et toutes sortes de logiciels malveillants. VirusTotal inspecte les éléments avec plus de 70 scanners antivirus et services de blocage d'URL/domaines, en plus d'une myriade d'outils pour extraire des signaux du contenu étudié. Tout utilisateur peut sélectionner un fichier sur son ordinateur à l'aide de son navigateur et l'envoyer à VirusTotal. VirusTotal propose plusieurs méthodes de soumission de fichiers, dont l'interface web publique principale, des téléchargeurs de bureau, des extensions de navigateur et une API . L'interface web a la priorité d'analyse la plus élevée parmi les méthodes de soumission publiques. Les soumissions peuvent être écrites dans n'importe quel langage de programmation en utilisant l'API publique basée sur HTTP. **Voici une liste** des sociétés qui participent à VirusTotal avec leur moteur antivirus :

[1] Système de gestion de base de données relationnelle.

[2] Une interface pour les programmes utilisant la plateforme Java. Elle permet aux applications Java d'accéder par le biais d'une interface commune à des sources de données pour lesquelles il existe des pilotes JDBC.



- ALYac
- Acronis (Acronis)
- AegisLab (AegisLab)
- Agnitum (Agnitum)
- AhnLab (V3)
- Alibaba Group (Alibaba)
- Antiy Labs (Antiy-AVL)
- Avast Software (Avast, Avast Mobile Security)
- Arcabit (Arcabit)
- AVG Technologies (AVG)
- Avira (AntiVir)
- Babable (Babable)
- BluePex (AVware)
- Baidu (Baidu-International)
- BitDefender GmbH (BitDefender)
- Bkav Corporation (Bkav)
- ByteHero Information Security Technology Team (ByteHero)
- Cat Computer Services (Quick Heal)
- Check Point Software Technologies (ZoneAlarm by Check Point)
- ClamAV (ClamAV)
- CMC InfoSec (CMC Antivirus)
- Comodo (Comodo)
- Cybereason (Cybereason)
- Cylance (Cylance)
- Cynet (Cynet)
- Cyren (Cyren)
- CrowdStrike (CrowdStrike Falcon (ML))
- Doctor Web, Ltd. (DrWeb)
- TEHTRI-Security (eGambit)
- Elasticsearch (Elastic)
- ESTsecurity (ALYac)
- Emsisoft Ltd (Emsisoft)
- Eset Software (ESET NOD32)
- FireEye (Fireeye)
- Fortinet (Fortinet)
- FRISK Software (F-Prot)
- F-Secure (F-Secure)
- G DATA Software (GData)
- Gridinsoft (Gridinsoft Antimalware Neural Network)
- Hacksoft (The Hacker)
- Hauri (ViRobot)
- IKARUS Security Software (IKARUS)
- Invincea (X by Invincea)
- INCA Internet (TACHYON)
- Jiangmin
- K7 Computing (K7AntiVirus, K7GW)
- Kaspersky Lab (Kaspersky)
- Kingsoft (Kingsoft)
- Lavasoft (Ad-Aware)
- Malwarebytes Corporation (Malwarebytes Anti-malware)
- Intel Security (McAfee)
- MAX (SaintSecurity)
- MaxSecure (MaxSecure)
- Microsoft (Malware Protection)
- Microworld (eScan)
- Nano Security (Nano Antivirus)
- Palo Alto Networks (Palo Alto Networks (Known Signatures))
- Panda Security (Panda Platinum)
- Qihoo 360 (Qihoo 360)
- Rising Antivirus (Rising)
- Sangfor (Sangfor)
- SentinelOne (SentinelOne (Static ML))
- Sophos (SAV)
- SUPERAntiSpyware (SUPERAntiSpyware)
- Symantec Corporation (Symantec, Symantec Mobile Insight)
- Tencent (Tencent)
- ThreatTrack Security (VIPRE Antivirus)
- TotalDefense (TotalDefense)
- Trapmine (Trapmine)
- Trend Micro (TrendMicro, TrendMicro-HouseCall)
- Trustlook (Trustlook Antivirus)
- VirusBlokAda (VBA32)
- Webroot (Webroot)
- Zillya! (Zillya)
- Zoner Software (Zoner Antivirus)

Figure 11. *Sociétés qui participent a VirusTotal.*

4.5 Conception d'un modèle ontologique de la hiérarchie des concepts

Dans cette étape, nous construisons le modèle de la hiérarchie des concepts. La hiérarchie de classification de concepts démontre l'organisation des concepts de l'ontologie dans un ordre hiérarchique qui exprime les relations sous-classe. Un concept universel « Thing », qui généralise tous les concepts racines des différentes hiérarchies de concepts est utilisé pour former une seule hiérarchie globale. Pour construire la taxonomie des concepts, Methontology[3] propose d'utiliser les quatre relations : Subclass-Of, Disjoint-Décomposition, Exhaustive-Décomposition, et Partition [80].

- Un concept C1 est une sous-classe de concept C2 si et seulement si toute instance de C1 est une instance de C2;
- Une Disjoint-Décomposition d'un concept C est un ensemble de sous-classes de C qui ne couvrent pas C et n'ont pas des instances communes;
- Une Exhaustive-Décomposition d'un concept C est un ensemble de sous-classes de C qui couvrent C et peuvent avoir des instances communes;
- Une partition d'un concept C est un ensemble de sous-classes de C qui couvrent C et n'ont aucune instance commune.

[3]Une méthodologie bien structurée pour construire des ontologies à partir de zéro. La méthodologie comprend un ensemble d'activités, de techniques.

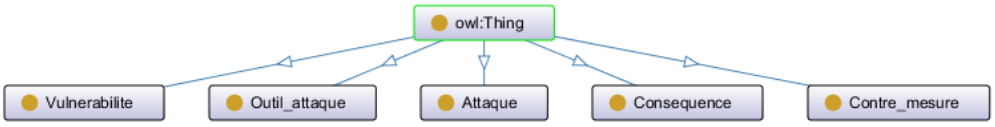


Figure 12. *Modèle ontologique de détection.*

- **Vulnérabilite** : Ce concept contient les différentes failles qu'un attaquant peut exploiter comme par exemple une faille dans le code PHP;
- **Outil_attaque** : Ce concept contient les noms des outils qu'un attaquant peut utiliser pour procéder à une attaque par exemple **Brutus** qui est un logiciel de brute force.[4];
- **Attaque** : Ce concept contient toutes les signatures des attaques;
- **Consequence** : Ce concept contient les dégâts laissé après une attaque;
- **Contre_mesure** : Ce concept contient les mesures a appliquer pour éviter une attaque.

Chaque concept possède une hiérarchie en appliquant les règles imposées par Methontology. Dans ce qui suit, nous allons présenter les hiérarchies des concepts sous-concept de Thing qui sont : **Attaque**, **Outil_attaque**, **Conséquence**, **Vulnerabilite** et **Contre_Mesure**.

[4]L'attaque par force brute est une méthode utilisée en cryptanalyse pour trouver un mot de passe ou une clé. Il s'agit de tester, une à une, toutes les combinaisons possibles.

4.5.1 Modèle ontologique des attaques

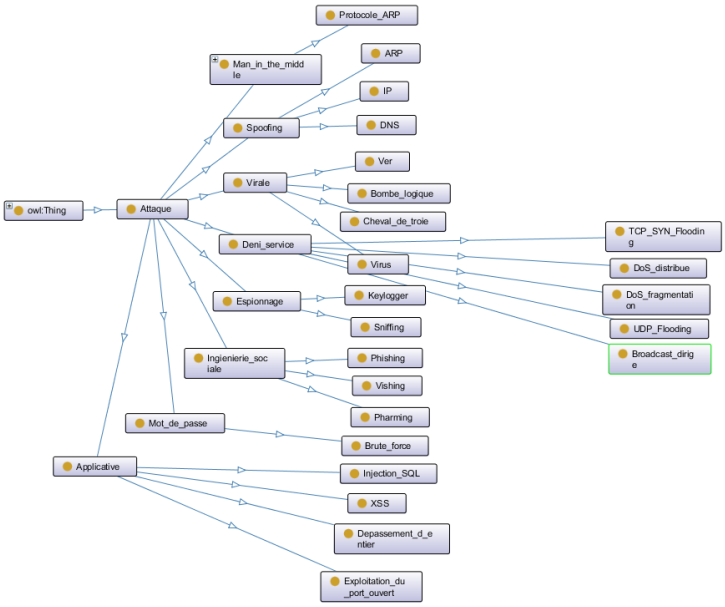


Figure 13. *Modèle ontologique des Attaques.*

Notre base de données ontologique permet de stocker les attaques par type 13, car grâce a **VirusTotal** nous connaîtrons tout les détails des attaques. Donc ces entités sont les différents types d'attaques stockés dans notre base de données ontologique.

4.5.2 Modèle ontologique des Vulnérabilités

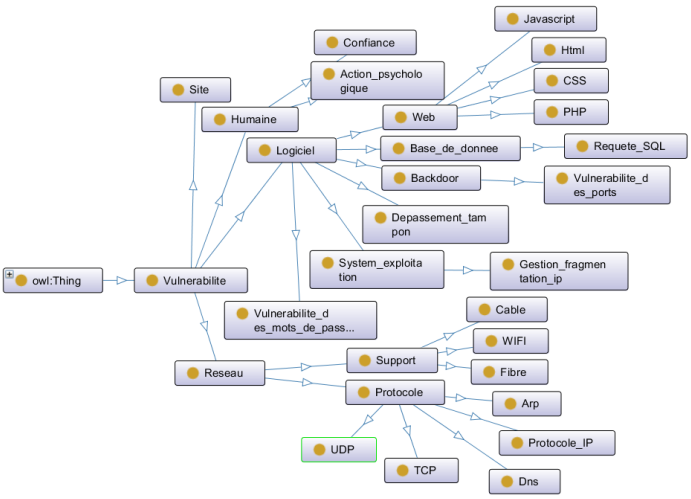


Figure 14. *Modèle ontologique des Vulnérabilités.*

Comme nous aurons tout les détails des attaques, nous allons connaître les failles qu'exploite chaque attaque que nous allons alors stocker dans notre base de données comme le montre la figure 14.

4.5.3 Modèle ontologique des Conséquences

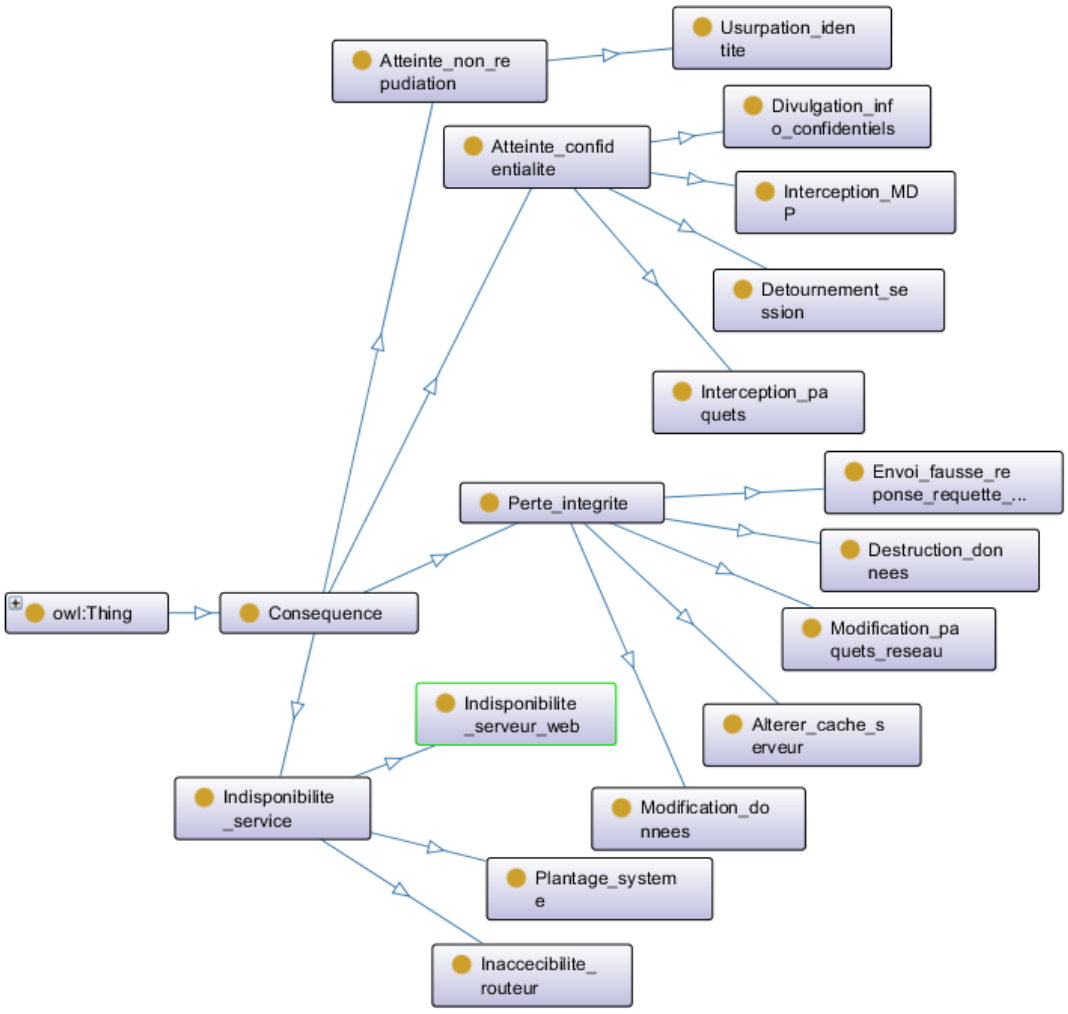


Figure 15. *Modèle ontologique des Conséquences.*

Grâce au concept Conséquence 15, le système de détection d'intrusions peut prévenir l'utilisateur de ce système l'endroit ou les dégâts on été causés par l'attaque.

4.5.4 Modèle ontologique des Outils d'attaques

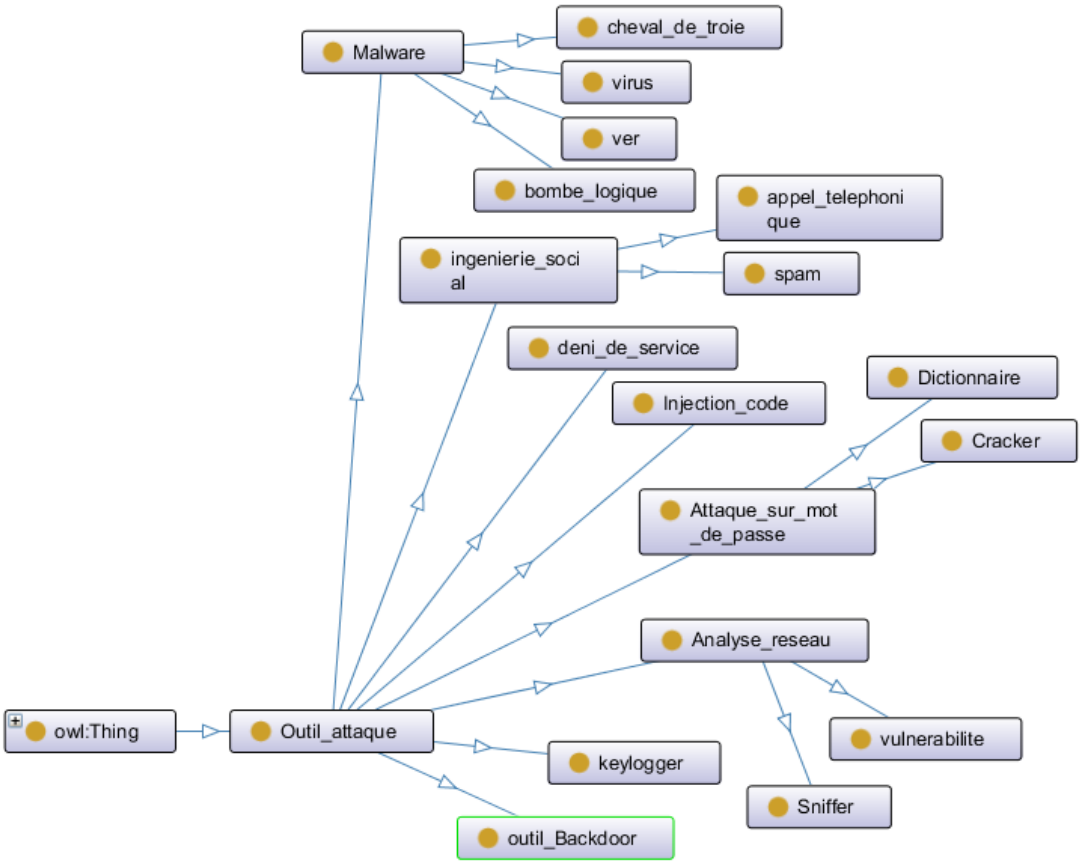


Figure 16. *Modèle ontologique des Outils d'attaques.*

Le concept outils d'attaque 16 va aider les chercheurs en cyber sécurité en leurs fournissant les outils utilisés par chaque attaque.

4.5.5 Modèle ontologique des Contre-mesures

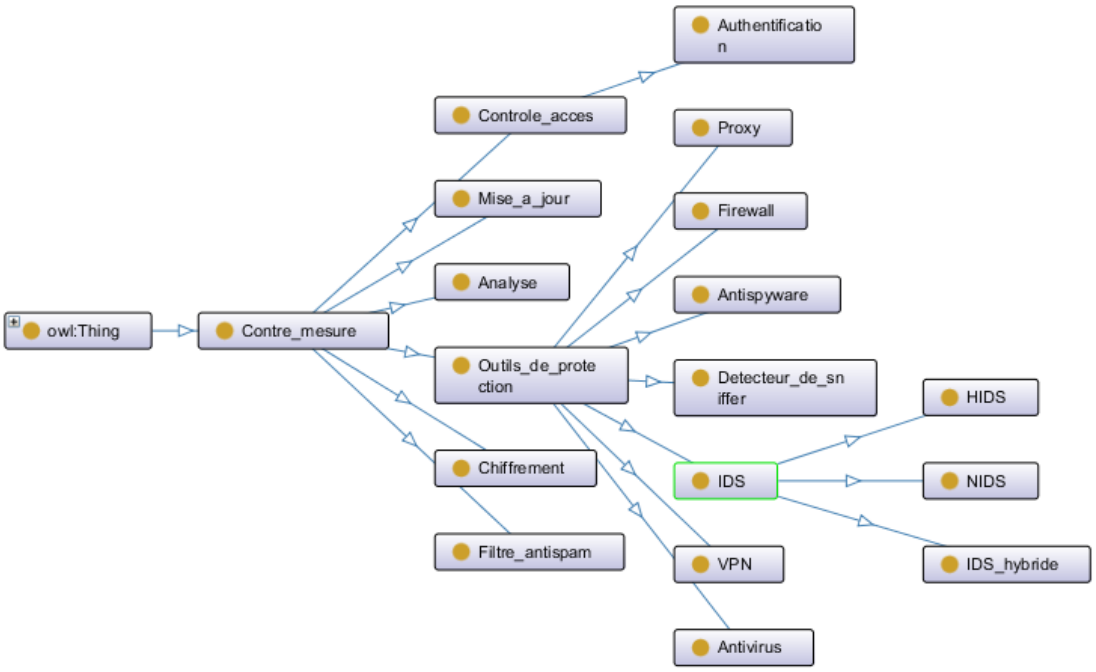


Figure 17. *Modèle ontologique des Contre-mesures.*

Les contres mesures mentionnées dans la Figure 17 représentent les solutions que le système propose a l'utilisateur pour améliorer la sécurité de son système.

4.6 L'ontologie du domaine d'application

Une fois que les concepts et leurs hiérarchies sont définies, il est nécessaire de relier tous les concepts en utilisant les relations sémantiques existantes. Ces relations qui existent entre nos concepts peuvent être définies de la manière comme illustré dans la Figure 18.

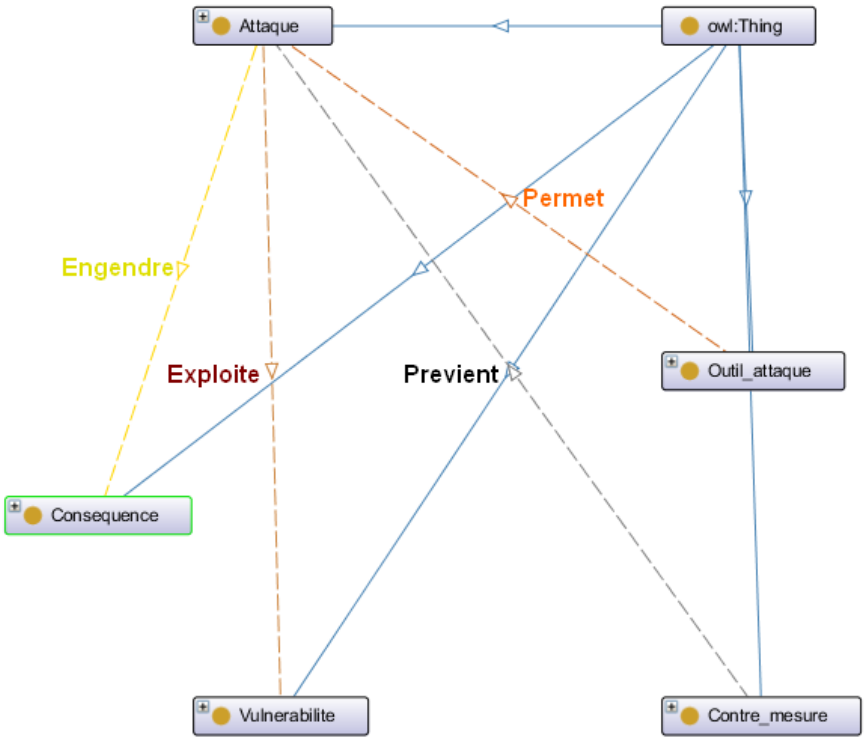


Figure 18. *Ontologie du domaine d'application.*

4.7 Les règles SWRL(Semantic Web Rule Language)

1. **Definition SWRL**
   SWRL est un langage de règles pour le web sémantique. Il est le résultat de la combinaison de OWL DL et OWL Lite et de RuleML. Il étend l'ensemble des axioms de OWL en y intégrant des clauses de Horn[5].

2. **Structure d'une règle SWRL**

```
a(?x, ?y) ^ b(?y, ?z) ^ c(?x) ^ . . . → n(?x, ?z)
```

[5]Les clauses de Horn sont ainsi nommées d'après le nom du logicien Alfred Horn qui est le premier à les avoir étudiées. Elles servent à garantir l'intégrité entre les requêtes et les bases de données ou de connaissances dans les systèmes évolués.

- **a, b, n :** prédicats binaires (rôles)
- **c :** prédicat unaire (concept atomique)
- **x, y, z :** variables, instances ou littéraux.

SWRL fournit une syntaxe abstraite de haut niveau en incluant dans les ontologies OWL des règles ayant une syntaxe abstraite[81].
Les règles définies par SWRL sont composées d'un antécédent (corp) et de la conséquence (en-tête), et si les conditions définies dans l'antécédent sont vérifiées alors les conditions définies dans la conséquence sont exécutées. Les deux parties constituant une règle (antécédent et conséquence) sont composées d'un ensemble d'atomes. Un antécédent ne contenant aucun atome est considéré comme vrai, c'est-à-dire qu'il est satisfait par toute interprétation, dans ce cas la conséquence est toujours satisfaite. Par contre une conséquence ne contenant aucun atome est considérée comme fausse et dans ce cas l'antécédent n'est vérifié par aucune interprétation.

3. **Écriture informelle d'une règle :**

```
parent(?x,?y) ^ frère(?y,?z) → oncle(?x,?z)
```

4. **Écriture abstraite d'une règle :**

```
Implies(Antecedent(hasParent(I-variable(x1) I-variable(x2))
hasBrother(I-variable(x2) I-variable(x3)))
Consequent(hasUncle(I-variable(x1) I-variable(x3))))
```

5. **Écriture d'une règle en XML, en ce basant sur la syntaxe d'OWL et RuleML :**

```
<swrlx:classAtom>
    <owlx:Class owlx:name="Person" />
    <ruleml:var>x1</ruleml:var>
</swrlx:classAtom>
<swrlx:classAtom>
    <owlx:IntersectionOf>
        <owlx:Class owlx:name="Person" />
        <owlx:ObjectRestriction owlx:property="hasParent">
            <owlx:someValuesFrom owlx:class="Physician" />
        </owlx:ObjectRestriction>
    </owlx:IntersectionOf>
    <ruleml:var>x2</ruleml:var>
</swrlx:classAtom>
```

4.8 Interrogation de l'ontologie

1. **Définition de SPARQL** : SPARQL est un langage d'interrogation à correspondance de graphes. Une requête SPARQL est de la forme H ← B, où B, le corps de la requête, est une expression de modèle de graphe RDF complexe qui peut inclure des triplets RDF avec des variables, des conjonctions, des disjonctions, des parties optionnelles et des contraintes sur les valeurs des variables, et H, la tête de la requête, est une expression qui indique comment construire la réponse à la requête. L'évaluation d'une requête Q par rapport à un ensemble de données D se fait en deux étapes : Le corps de Q est comparé à D afin d'obtenir un ensemble de liaisons pour les variables du corps, puis, en utilisant les informations sur la tête de Q, ces liaisons sont traitées en appliquant les opérateurs relationnels classiques (projection, distinct, etc.) afin de produire la réponse à la requête, qui peut prendre différentes formes, telles qu'une réponse oui/non, un tableau de valeurs ou un nouveau jeu de données RDF.
2. **Syntaxe SPARQL** : La syntaxe officielle de SPARQL considère les opérateurs OPTIONNELS, UNION, et FILTRE, et la concaténation via un symbole de point (.), pour construire des expressions de motifs de graphes. La syntaxe prend également en compte les pour regrouper les motifs, ainsi que certaines règles implicites de précédence et d'association. Par exemple, le symbole de point (.) a la priorité sur OPTIONAL, et OPTIONAL est associatif à gauche. Afin d'éviter les ambiguïtés dans l'analyse syntaxique, une présentation de la syntaxe des motifs de graphes SPARQL dans un formalisme algébrique plus traditionnel, est utilisée en exploitant les opérateurs binaires AND (.), UNION (UNION), OPT (OPTIONAL), et FILTER (FILTRE). Les expressions sont met entre parenthèses pour rendre explicites la préséance et l'association des opérateurs. Une expression de motif de graphe SPARQL est définie de manière récursive comme suit.

```
(1) Un tuple de (I u L u V ) × (I u V ) × (I u L u V )
est un motif de graphe (un motif triple).

(2) Si P1 et P2 sont des motifs de graphe,
alors les expressions (P1 AND P2), (P1 OPT P2),
et (P1 UNION P2) sont des motifs de graphe
(motif de graphe de conjonction, motif de graphe
optionnel, et motif de graphe d'union, respectivement).

(3) Si P est un motif de graphe et R est une condition
intégrée à SPARQL, alors l'expression (P FILTER R)
est un motif de graphe (un motif de graphe filtrant).
```

Une condition intégrée dans SPARQL est construite en utilisant des éléments de l'ensemble I u L u V et des constantes, des connecteurs logiques, des symboles d'inégalité, le symbole d'égalité, des prédicats unaires :

```
(1) Si ?X , ?Y appartient V et c appartient I u L, alors bound(
?X = c et ?X = ?Y sont des conditions intégrées.

(2) Si R1 et R2 sont des conditions intégrées, alors (NOT R1),
(R1 OR R2) et (R1 AND R2) sont des conditions intégrées.
```

4.9 Conclusion

Dans ce chapitre nous avons présenté en détail notre modele ontologique ainsi que les différents outils que nous avons utiliser pour concevoir notre politique de sécurité pour qu'elle puisse détecter, identifier et sauvegarder les signatures des cyber-attaques.

5. Chapitre V Contributions

5.1 Introduction

Dans ce chapitre nous avons détaillé chaque phase de l'approche, en commençant par la phase de **détection** suivie par la phase d'**identification** et enfin la phase de **sauvegarde**.

5.2 Simulation d'une attaque

Pour cette simulation nous allons utiliser un fichier contenant un cheval de troie (Trojan) qui est un logiciel en apparence légitime, mais qui contient une fonctionnalité malveillante. Nous allons télécharger le fichier dans notre "plateforme attaquable" afin de savoir comment notre serveur proxy détecte l'attaque, l'identifie, la bloque et la sauvegarde dans notre base de données ontologique.

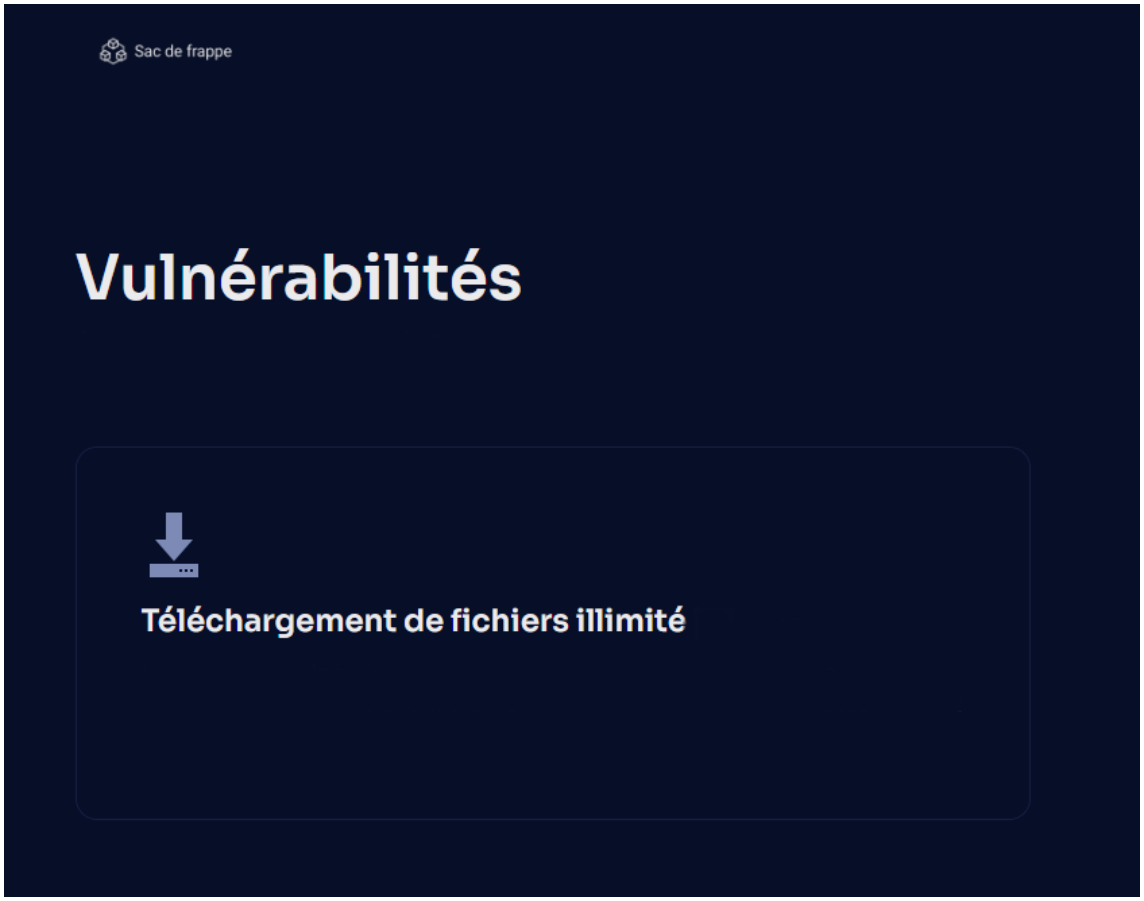


Figure 19. *Page d'accueil de la plateforme attaquable.*

5.2.1 Phase de détection

Pour la phase de détection notre serveur proxy hash en md5[1] le fichier télécharger sur notre plateforme attaquable et analyse ce dernier grâce a **VirusTotal** comme illustré dans la figure 19 pour le téléchagement et la figure 20 pour le hashage et c'est ce que nous allons expliquer dans le chapitre suivant.

```
a=hashlib.md5(flow.request.get_content().decode("utf-8")[x:y].encode('utf-8'))
```

MD5 dce172521b281347a1e0afe7ec4b1aa4

Figure 20. *Hash en md5 avec python.*

Upload d'un fichier malveillant sans notre proxy :

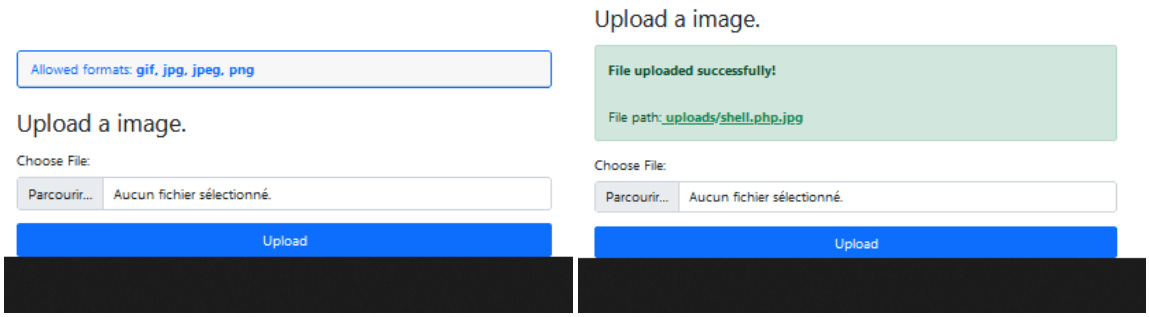


Figure 21. *Upload d'un fichier malveillant sans notre proxy.*

Nous remarquons que le fichier a été telecharger sans problèmes malgré le fait qu'il soit malveillant.

5.2.2 Phase d'identification

Pour la phase d'identification nous avons utilisé **VirusTotal**. Puisque VirusTotal analyse à la demande, nous l'utilisons comme base de données. C'est à dire que notre serveur proxy prend la signature du fichier envoyer par le client comme le montre la Figure 22, l'envoi à VirusTotal qui l'analyse et renvoi la réponse au proxy qui selon les résultats de l'analyse affichés dans le Figure 23:

- **Négatif :** Bloque la requête et enregistre la signature du fichier dans la base de données ontologique.

[1]Le MD5, pour Message Digest 5, est une fonction de hachage cryptographique qui permet d'obtenir l'empreinte numérique d'un fichier. Il a été inventé par Ronald Rivest en 1991

- **Positif :** Laisse passer la requête.

Page d'accueil :

Figure 22. *Page d'accueil VirusTotal.*

Résultats d'une analyse :

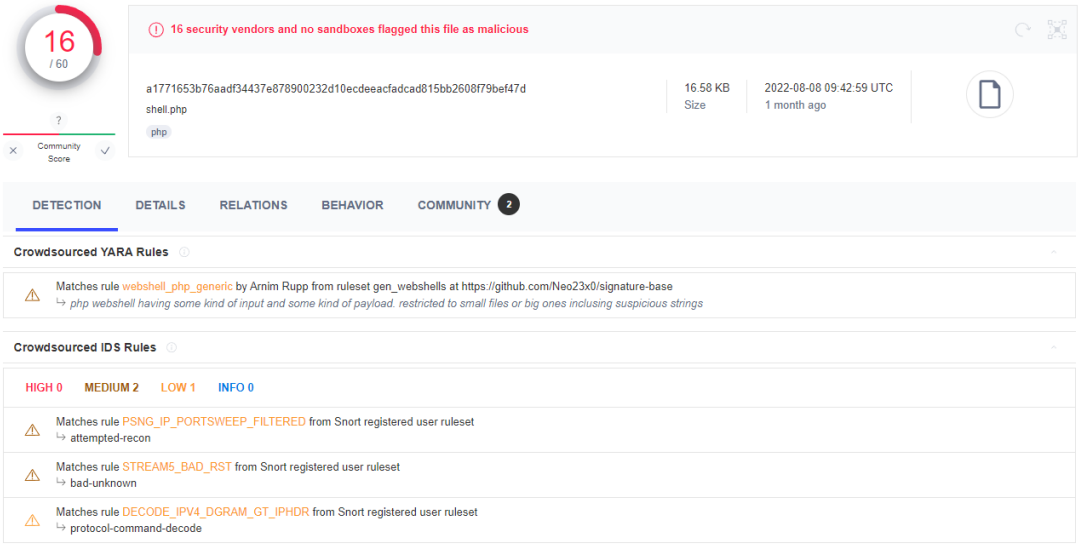
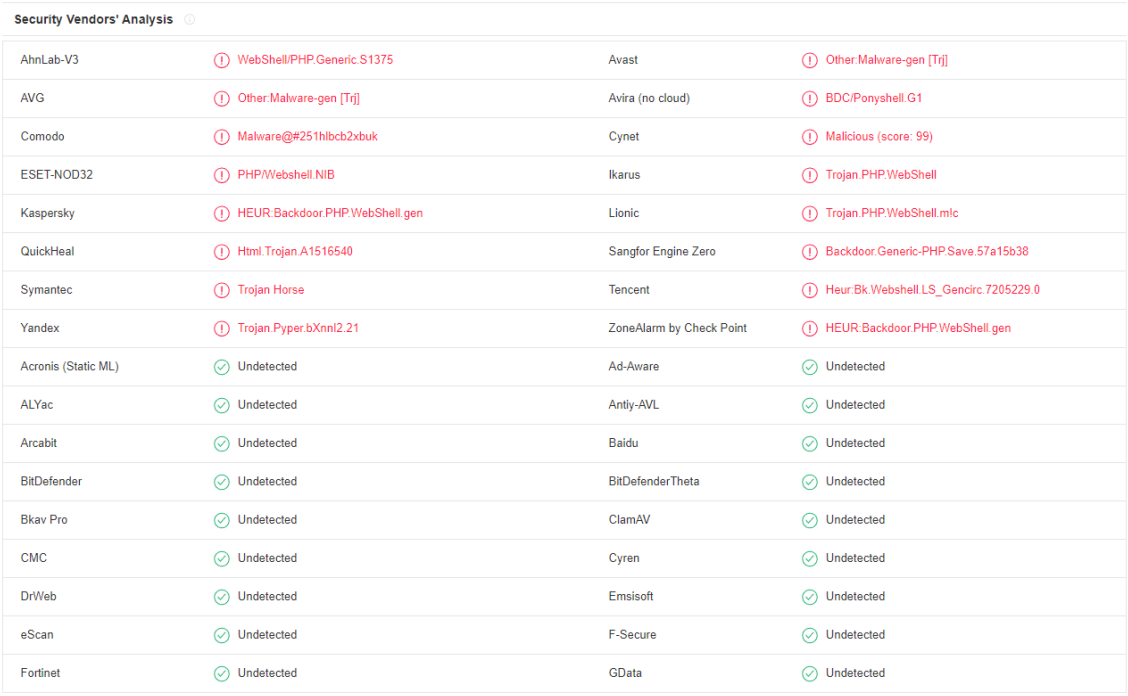


Figure 23. *Résultats d'analyse.*



Security Vendors' Analysis

AhnLab-V3	WebShell/PHP.Generic.S1375	Avast	Other:Malware-gen [Trj]
AVG	Other:Malware-gen [Trj]	Avira (no cloud)	BDC/Ponyshell.G1
Comodo	Malware@#251hlbcb2xbuk	Cynet	Malicious (score: 99)
ESET-NOD32	PHP/Webshell.NIB	Ikarus	Trojan.PHP.WebShell
Kaspersky	HEUR:Backdoor.PHP.WebShell.gen	Lionic	Trojan.PHP.WebShell.mlc
QuickHeal	Html.Trojan.A1516540	Sangfor Engine Zero	Backdoor.Generic-PHP.Save.57a15b38
Symantec	Trojan Horse	Tencent	Heur:Bk.Webshell.LS_Gencirc.7205229.0
Yandex	Trojan.Pyper.bXnnl2.21	ZoneAlarm by Check Point	HEUR:Backdoor.PHP.WebShell.gen
Acronis (Static ML)	Undetected	Ad-Aware	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
Arcabit	Undetected	Baidu	Undetected
BitDefender	Undetected	BitDefenderTheta	Undetected
Bkav Pro	Undetected	ClamAV	Undetected
CMC	Undetected	Cyren	Undetected
DrWeb	Undetected	Emsisoft	Undetected
eScan	Undetected	F-Secure	Undetected
Fortinet	Undetected	GData	Undetected

Figure 24. *Résultats d'analyse des moteurs antivirus.*

Après cette analyse VirusTotal mentionne que le fichier est malicieux, dans ce cas, notre serveur proxy le bloquera et enregistrera sa signature dans notre base de données ontologique.

VirusTotal nous offre l'avantage d'avoir plusieurs moteurs d'antivirus qui peuvent analyser notre fichier, et ce qui permet l'amélioration de la précision de détection, comme dans notre cas le moteur **Fortinet** ne détecte aucune anomalie dans le fichier, tandis que le moteur **Kaspersky** détecte un cheval de troie (Trojan) dans le fichier.

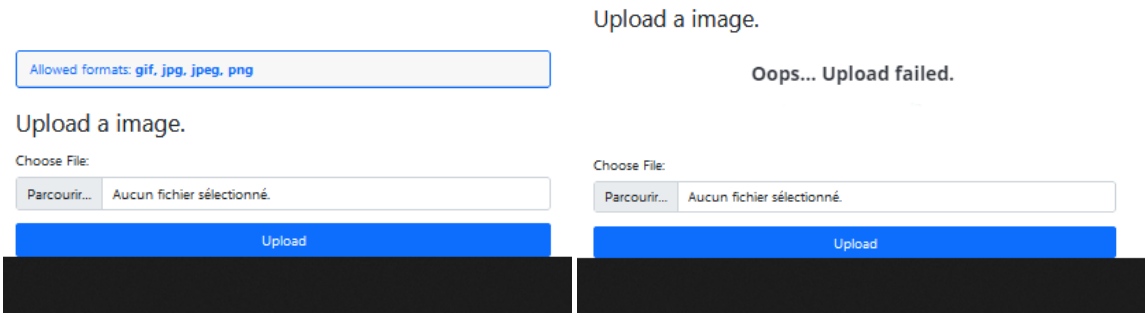


Figure 25. *Upload d'un fichier malveillant avec notre proxy.*

Après l'activation de notre proxy, le fichier a été bloqué et n'a pas pu être télécharger.

5.2.3 Phase de sauvegarde

Dans cette phase, notre proxy va sauvegarder les signatures des cyber-attaques dans la base de données ontologique. afin que nous puissions effectué du raisonnement à base des règles d'inférences en utilisant ces données ontologique.

5.2.3.1 Définition des instances

Nous avons instancié les différents concepts et les relations entre celles-ci, ensuite nous avons attribué des valeurs aux différents attributs.

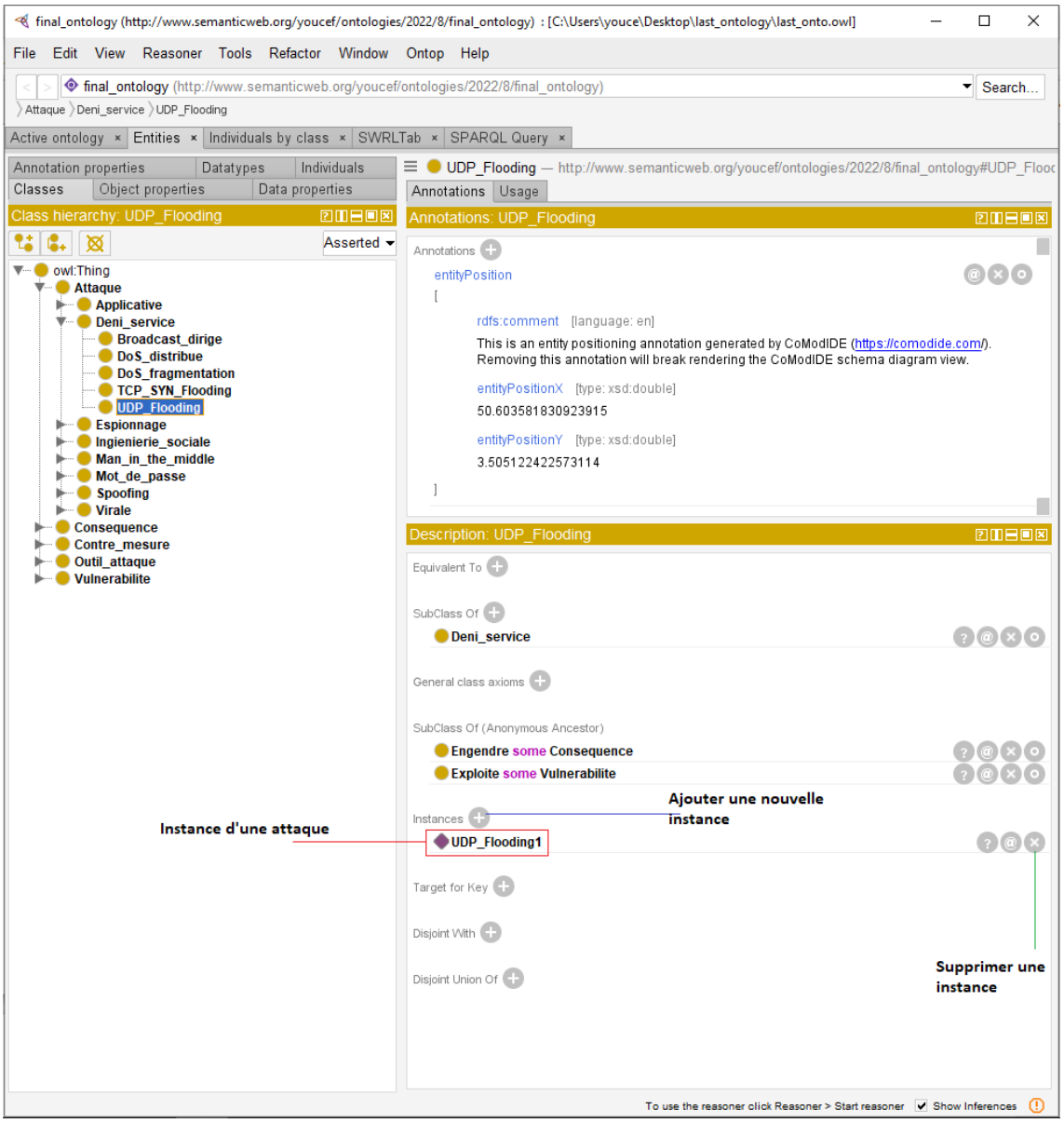


Figure 26. *Définition d'une instance.*

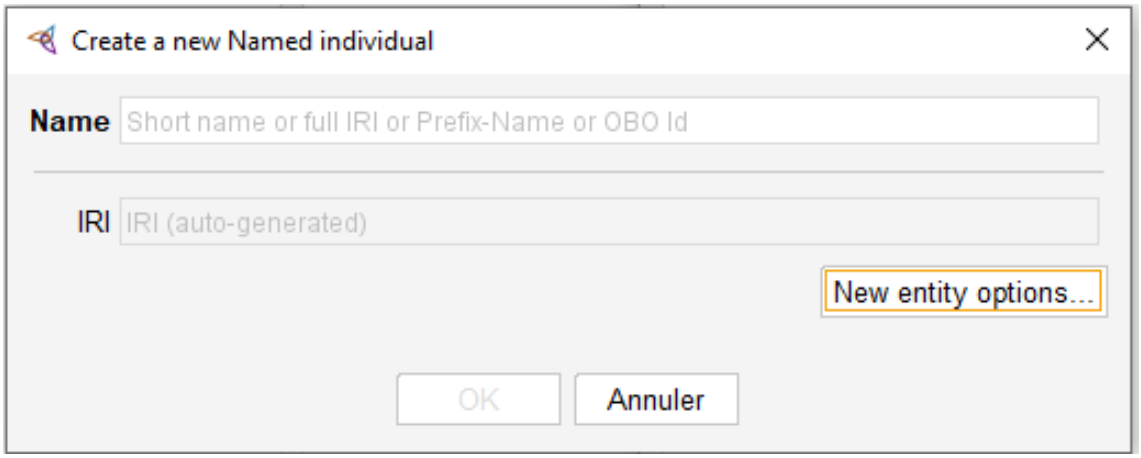


Figure 27. *Fenêtre d'ajout d'une instance.*

5.2.3.2 Implémenter des règles dans notre base de données :

Après la conception d'une ontologie consistante avec les instances de données nécessaires, nous avons implémenté des règles SWRL(Semantic Web Rule Language) qui est un langage de règles pour le web sémantique afin de pouvoir effectuer des inférences, et déduire de nouvelles relations et/ou données sémantiques à partir de celles que nous avons (relations et/ou données primitives). Pour cela, nous avons inférer de nouvelles relations à base de règles d'inférences comme définit dans la Figure 29 et les règles ci-après.

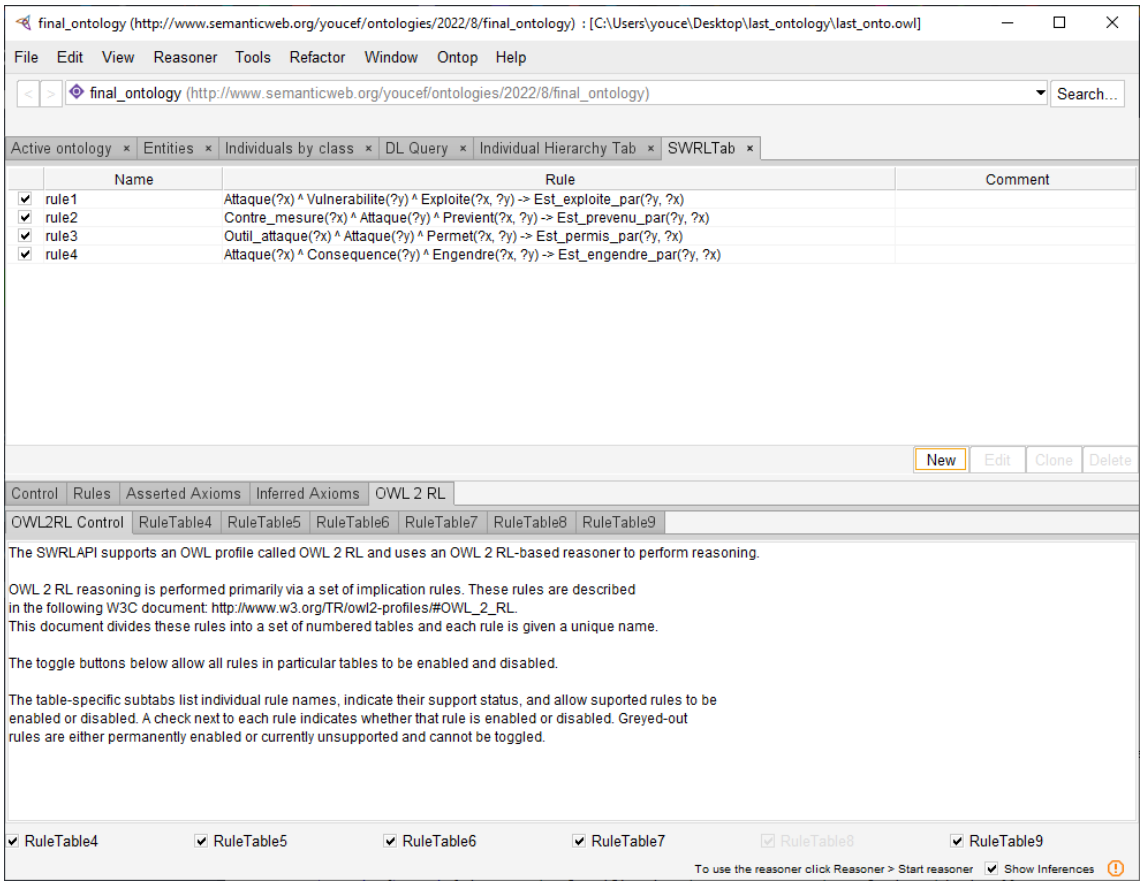


Figure 28. *Implémentation des règles dans protégé*

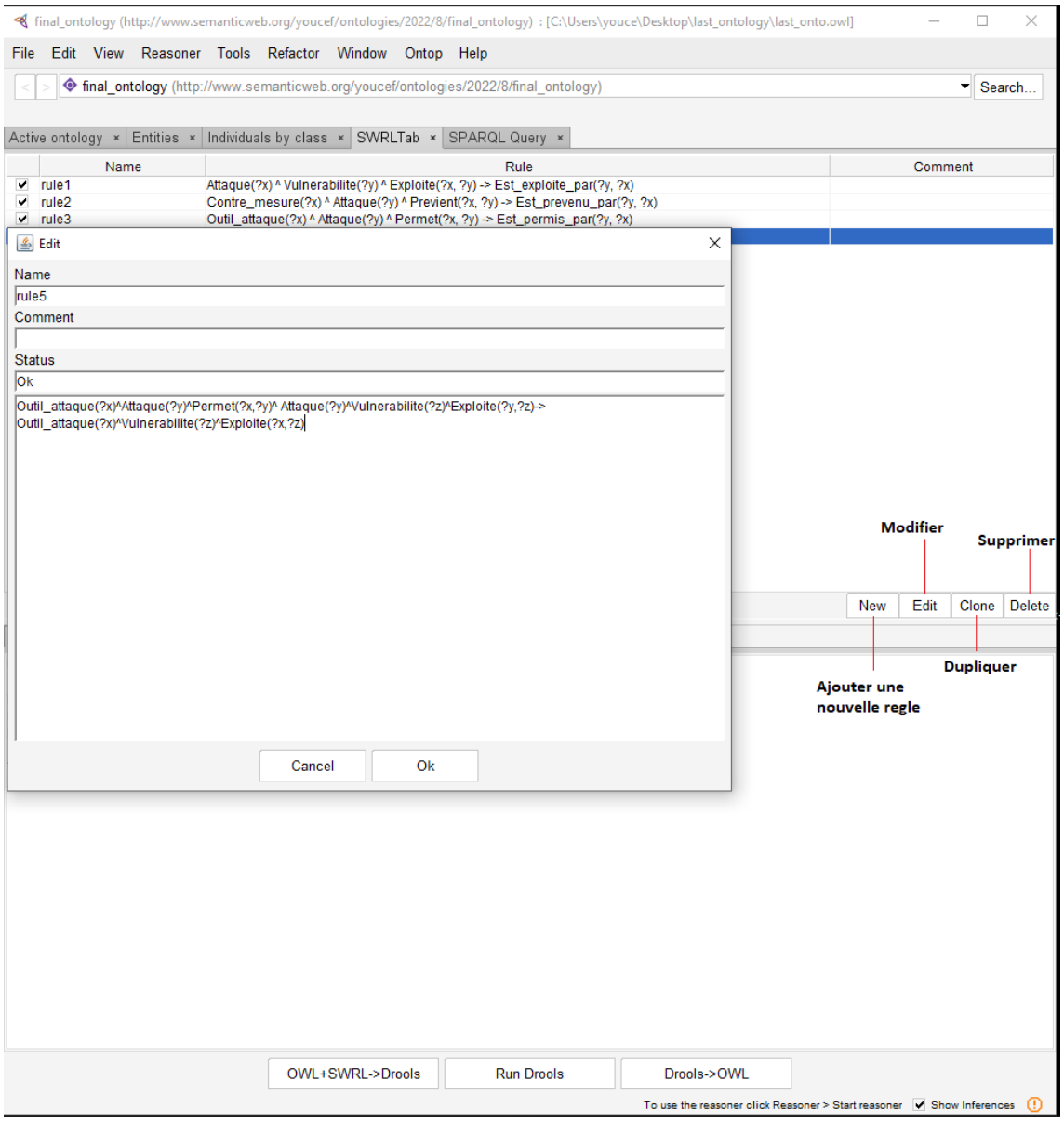


Figure 29. *Fenêtre d'ajout d'une règle.*

- **Rule 1:**

```
Attaque(?x)^Vulnerabilite(?y)^Exploite(?x,?y)
->Est_exploite_par(?y,?x)
```

Si une attaque exploite une vulnerabilite alors la vulnerabilite est exploiter par l'attaque.

- **Rule 2:**

```
Contre_mesure(?x)^Attaque(?y)^Previent(?x,?y)
->Est_prevenu_par(?y,?x)
```

Si une Contre mesure Previent une Attaque alors l'attaque est prevenue par la contre mesure.

- **Rule 3:**

```
Outil_attaque(?x)^Attaque(?y)^Permet(?x,?y)
->Est_permis_par(?y,?x)
```

Si un outil d'attaque permet une attaque alors l'attaque est permis par l'outil d'attaque.

- **Rule 4:**

```
Attaque(?x)^Consequence(?y)^Engendre(?x,?y)
->Est_engendre_par(?y,?x)
```

Si une attaque Engendre une consequence alors la consequence est engendre par l'attaque.

- **Rule 5:**

```
Outil_attaque(?x)^Attaque(?y)^Permet(?x,?y)
^ Attaque(?y)^Vulnerabilite(?z)^Exploite(?y,?z)
-> Outil_attaque(?x)^Vulnerabilite(?z)^Exploite(?x,?z)
```

Si un outil d'attaque permet une attaque, et cette attaque exploite une vulnérabilité, alors cette outil d'attaque exploite cette vulnérabilité.

5.2.3.3 L'interrogation de l'ontologie en utilisant le langage SPARQL sur SPARQL Query Panel

Protégé propose un Plugin qui permet d'exécuter des requêtes SPARQL directement sur l'ontologie. C'est « SPARQL Query Panel ». Nous avons exécuté des requêtes en utilisant le langage SPARQL.

1. Cette requête permet de récupérer tous les concepts qui sont reliés par la relation « SubClasseOf », la relation de subsomption ou d'héritage.

```
SELECT ?subject ?object
WHERE { ?subject rdfs:subClassOf ?object }
```

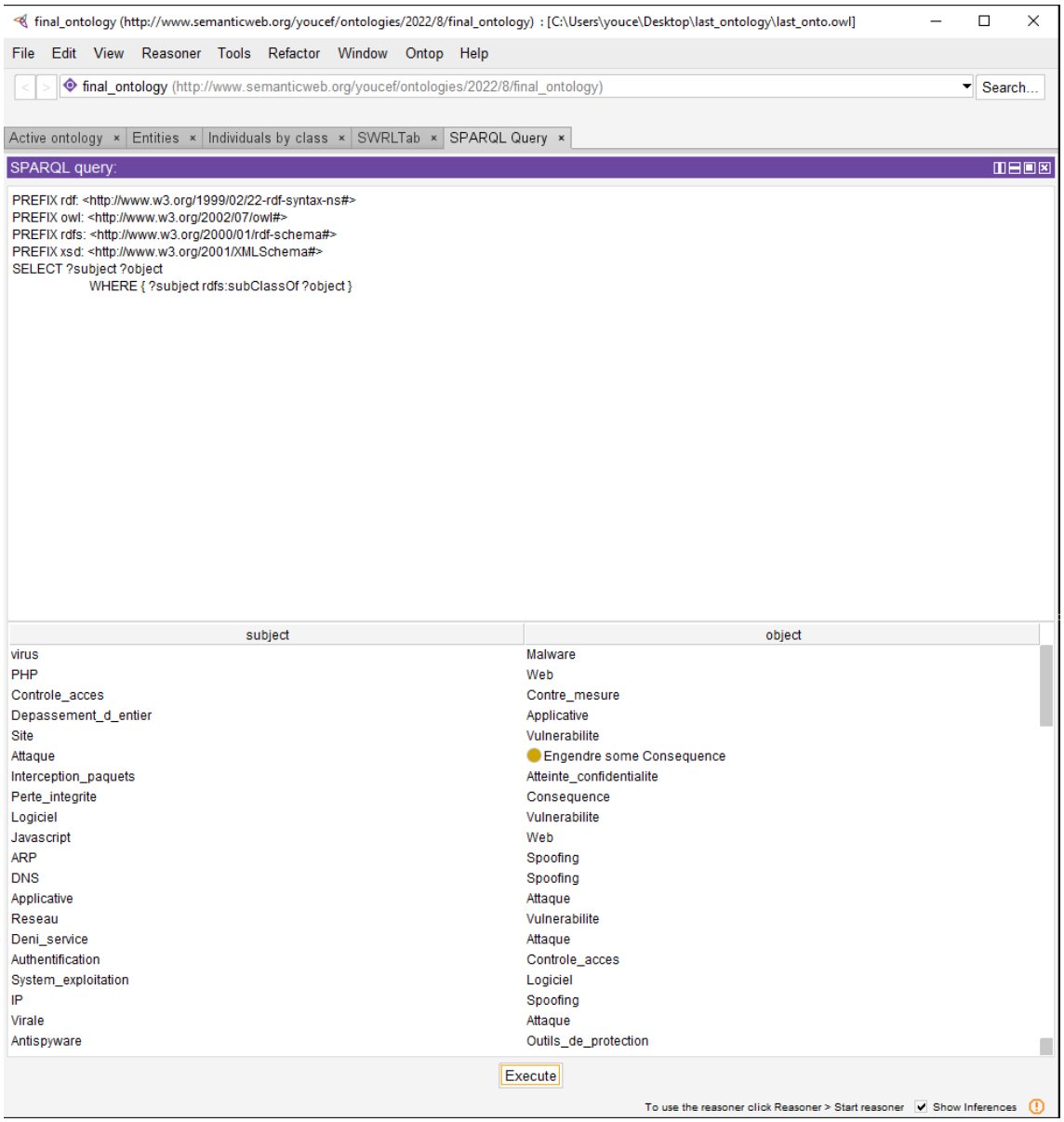


Figure 30. *Résultat de la requête 1.*

2. Cette requête permet de récupérer toutes les instances qui sont reliés par la relation «Exploite».

```
SELECT ?subject ?object
WHERE { ?subject :Exploite ?object }
```

3. Cette requête permet de donner les contre-mesures qui préviennent de chaque attaque.

```
SELECT ?object ?subject
WHERE { ?subject :Previent ?object }
```

Dans cette section nous avons présenté en détail l'implémentation de notre ontologie. Nous sommes passés d'une ontologie formelle à une ontologie consistante. Pour cela nous avons utilisé l'éditeur d'ontologie Protégé. Nous avons d'abord commencé par la définition des concepts, attributs et relations. Puis nous avons défini toutes les instances de notre domaine et nous les avons reliées entre elles en instanciant les relations définies précédemment. Ensuite, nous avons définis des règles SWRL afin d'effectuer des inférences et déduire de nouvelles connaissances qui sont des données sémantiques.

5.3 Conclusion

Dans ce chapitre nous avons détaillé chaque phase de l'approche, en commençant par la phase de détection qui dépend de l'interception des requêtes par notre serveur proxy celui ci va hash en md5 le contenue de la requête et l'envoyer a **VirusTotal**, ce dernier entame la phase d'identification qui consiste a scanner le contenue de la requête grâce a ses moteurs antivirus et envoyer le résultat du scan au proxy, si les résultats sont positifs et que le contenue de la requête est sans danger alors le proxy laisse passer la requête sans problèmes, sinon si les résultats du scan sont négatifs et que le contenue de la requête est malveillant alors le proxy entame la phase de sauvegarde, dans cette phase le proxy sauvegarde la signature du contenue de la requête dans notre base de données ontologique afin que nous puissions effectué du raisonnement à base des règles d'inférences en utilisant ces données ontologique.

Conclusion générale

5.4 Conclusion générale

Notre travail s'insère dans le domaine de la sécurité informatique. Le but de notre travail est de rendre les systèmes de détection d'intrusions plus efficace en proposant une approche à base d'ontologie afin de diminuer les fausses alertes.

Pour ce faire, nous avons commencé par introduire les systèmes de détection d'intrusions. Nous avons présenté par le suite le concept d'ontologie. Enfin, nous avons montré le fonctionnent des systèmes de détection d'intrusions à base d'ontologies.

La contribution principale de notre travail est de concevoir, d'utiliser et d'exploiter un modèle ontologique contenant des classes de base et des sous-classes pour les concepts de notre contexte de travail, d'alerte, de vulnérabilité et d'attaque. Ces ontologies sont alors peuplées soit automatiquement par les pilotes de source spécifique (Notre serveur proxy est une de ces sources), ou manuellement pour les données statiques.

Pour ce faire nous avons conçu un serveur proxy qui détecte, identifie et sauvegarde les cyber-attaques dans notre base de données ontologique. Ce dernier permet l'élaboration d'un système de détection d'intrusions à base d'ontologie de domaine. Ce proxy vérifié chaque requête venant de l'extérieur du serveur et chaque requête sortante du serveur client afin d'assurer une vérification extrême et détaillée.

5.5 Perspectives

Nous projetons notre système de détection d'intrusions à base d'ontologie dans le domaine du machine learning et/ou deep learning, car récemment, des systèmes IDS basés sur l'apprentissage automatique (ML) et l'apprentissage profond (DL) ont été déployés comme solutions potentielles pour détecter efficacement les intrusions sur les réseaux.

Références

[1] Karthik Kannan and Rahul Telang. “Market for software vulnerabilities? Think again”. In: *Management science* 51.5 (2005), pp. 726–740.

[2] Sam Ransbotham, Sabyaschi Mitra, and Jon Ramsey. “Are markets for vulnerabilities effective?” In: *Mis Quarterly* (2012), pp. 43–64.

[3] Carl Sabottke, Octavian Suciu, and Tudor Dumitraș. “Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting {Real-World} Exploits”. In: *24th USENIX Security Symposium (USENIX Security 15)*. 2015, pp. 1041–1056.

[4] Pauline Bowen, Joan Hash, and Mark Wilson. “Information security handbook: a guide for managers”. In: *NIST special publication 800-100, National Institute of Standards and Technology*. Citeseer. 2007.

[5] Stefan Fenz and Andreas Ekelhart. “Formalizing information security knowledge”. In: *Proceedings of the 4th international Symposium on information, Computer, and Communications Security*. 2009, pp. 183–194.

[6] Raphael Yende. “SUPPORT DE COURS DE SÉCURITÉ INFORMATIQUE ET CRYPTO.” In: (2018).

[7] James P Anderson. “Computer security threat monitoring and surveillance”. In: *Technical Report, James P. Anderson Company* (1980).

[8] Dorothy E Denning. “An intrusion-detection model”. In: *IEEE Transactions on software engineering* 2 (1987), pp. 222–232.

[9] Philipe Biondi. “Architecture expérimentale pour la détection d’intrusions dans un système informatique”. In: *Article de recherche, Avril-Sptembre* (2001).

[10] Richard Heady et al. *The architecture of a network level intrusion detection system*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States); New Mexico …, 1990.

[11] Serge Fenet and Salima Hassas. “A distributed Intrusion Detection and Response System based on mobile autonomous agents using social insects communication paradigm”. In: *Electronic Notes in Theoretical Computer Science* 63 (2002), pp. 41–58.

[12] Karima Boudaoud. “Détection d’intrusions: Une nouvelle approche par systèmes multi-agents”. PhD thesis. Verlag nicht ermittelbar, 2002.

[13] Nicolas Nobelis. “Un modèle de Case-Based Reasoning pour la détection d’intrusion”. In: *Rapport de stage DEA RSD/ESSI3 SAR* (2004).

[14] Hervé Debar and Andreas Wespi. “Aggregation and correlation of intrusion-detection alerts”. In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2001, pp. 85–103.

[15] Phillip A Porras and Alfonso Valdes. “Live Traffic Analysis of TCP/IP Gateways.” In: *NDSS*. 1998.

[16] Kenza Tayeb. “Détection d’intrusion coopérative basée sur la fusion de données”. PhD thesis. Thèse de Magister de l’INI, 2006.

[17] Hervé Debar, Marc Dacier, and Andreas Wespi. “A revised taxonomy for intrusion-detection systems”. In: *Annales des télécommunications*. Vol. 55. 7. Springer. 2000, pp. 361–378.

[18] Boussad Ait Salem and Mourad Brahimi. *Conception et réalisation d’un système de détection d’intrusions*. 2005.

[19] Ludovic Me and Cédric Michel. “La détection d’intrusions: bref aperçu et derniers développements”. In: *Mars* (1999).

[20] Jean-Marc Percher and Bernard Jouga. “Détection d’intrusions dans les réseaux ad hoc”. In: *Projet* (2003).

[21] Ludovic Mé—Véronique Alanou. “Détection d’intrusion dans un systeme informatique: méthodes et outils”. In: *SUPELEC BP* 2835511 (1996).

[22] David Burgermeister and Jonathan Krier. *Les systèmes de détection d’intrusions*. 2006.

[23] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. “Cyber scanning: a comprehensive survey”. In: *Ieee communications surveys & tutorials* 16.3 (2013), pp. 1496–1519.

[24] Se-Yul Lee et al. “A probe detection model using the analysis of the fuzzy cognitive maps”. In: *International Conference on Computational Science and its Applications*. Springer. 2005, pp. 320–328.

[25] Hung Nguyen Viet et al. “Using deep learning model for network scanning detection”. In: *Proceedings of the 4th International Conference on Frontiers of Educational Technologies*. 2018, pp. 117–121.

[26] Prakash Veeraraghavan, Dalal Hanna, and Eric Pardede. “NAT++: an efficient micro-nat architecture for solving ip-spoofing attacks in a corporate network”. In: *Electronics* 9.9 (2020), p. 1510.

[27] Jing Xia et al. “An active defense solution for ARP spoofing in OpenFlow network”. In: *Chinese Journal of Electronics* 28.1 (2019), pp. 172–178.

[28] Mykola Kozlenko and Valerii Tkachuk. “Deep learning based detection of DNS spoofing attack”. In: (2019).

[29] Ahmed Sheikh. *Certified Ethical Hacker (CEH) Preparation Guide*. Springer, 2021.

[30] Farouq Aliyu, Tarek Sheltami, and Elhadi M Shakshuki. "A detection and prevention technique for man in the middle attack in fog computing". In: *Procedia computer science* 141 (2018), pp. 24–31.

[31] IV Kotenko and AV Ulanov. "Agent-based simulation of DDOS attacks and defense mechanisms". In: *Journal of Computing* 4.2 (2005), pp. 16–37.

[32] Robert Neches et al. "Enabling technology for knowledge sharing". In: *AI magazine* 12.3 (1991), pp. 36–36.

[33] Thomas R Gruber. "A translation approach to portable ontology specifications". In: *Knowledge acquisition* 5.2 (1993), pp. 199–220.

[34] KAIS SALHI. "LA FUSION DES ONTOLOGIES". In: (2014).

[35] Thomas R Gruber. "Toward principles for the design of ontologies used for knowledge sharing?" In: *International journal of human-computer studies* 43.5-6 (1995), pp. 907–928.

[36] Willem Nico Borst. "Construction of engineering ontologies for knowledge sharing and reuse." In: (1999).

[37] Michael Grüninger and Mark S Fox. "Methodology for the design and evaluation of ontologies". In: (1995).

[38] Robert B Allen. "Semantic modeling with SUMO". In: *arXiv preprint arXiv:2012.15835* (2020).

[39] Audrey Baneyx. "Construire une ontologie de la Pneumologie Aspects théoriques, modèles et expérimentations". PhD thesis. Université Pierre et Marie Curie-Paris VI, 2007.

[40] Pablo Beltrán-Ferruz, Pedro A González-Calero, and Pablo Gervás. "Converting Mikrokosmos frames into description logics". In: *Proceeedings of the Workshop on NLP and XML (NLPXML-2004): RDF/RDFS and OWL in Language Technology*. 2004, pp. 35–42.

[41] Moh Muklis Sulaeman and Mugi Harsono. "Supply Chain Ontology: Model Overview and Synthesis". In: *Jurnal Mantik* 5.2 (2021), pp. 790–799.

[42] Espinasse Bernard. "Introduction aux Ontologies". In: *Professeur à l'Université d'Aix-Marseille* (2010).

[43] Dave Beckett. "Rdf/xml syntax specification w3c recommendation". In: *http://www. w3. org/TR/rdf-syntax-grammar/* (2004).

[44] Dan Brickley, Ramanathan V Guha, and Brian McBride. "RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation (2004)". In: *URL http://www. w3. org/tr/2004/rec-rdf-schema* 20040210 (2004).

[45] Sean Bechhofer. “OWL web ontology language reference, W3C Recommendation”. In: *http://www. w3. org/TR/owl-ref/* (2004).

[46] Ravi S Sandhu et al. “Role-based access control models”. In: *Computer* 29.2 (1996), pp. 38–47.

[47] Wei-Tek Tsai and Qihong Shao. “Role-based access-control using reference ontology in clouds”. In: *2011 Tenth International Symposium on Autonomous Decentralized Systems*. IEEE. 2011, pp. 121–128.

[48] Gail-Joon Ahn and Ravi Sandhu. “Role-based authorization constraints specification”. In: *ACM Transactions on Information and System Security (TISSEC)* 3.4 (2000), pp. 207–226.

[49] Andrzej Uszok et al. “KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement”. In: *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*. IEEE. 2003, pp. 93–96.

[50] Lalana Kagal, Tim Finin, and Anupam Joshi. “A policy language for a pervasive computing environment”. In: *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*. IEEE. 2003, pp. 63–74.

[51] Lalana Kagal et al. “Rei: A policy language for the me-centric project”. In: (2002).

[52] Dan Constantin Tofan. “Information security standards”. In: *Journal of Mobile, Embedded and Distributed Systems* 3.3 (2011), pp. 128–135.

[53] Robert Shirey. *Internet security glossary*. Tech. rep. 2000.

[54] Matthew A Bishop. *The art and science of computer security*. 2002.

[55] Almut Herzog, Nahid Shahmehri, and Claudiu Duma. “An ontology of information security”. In: *International Journal of Information Security and Privacy (IJISP)* 1.4 (2007), pp. 1–23.

[56] Ju An Wang, Michael M Guo, and Jairo Camargo. “An ontological approach to computer system security”. In: *Information Security Journal: A Global Perspective* 19.2 (2010), pp. 61–73.

[57] Ju An Wang and Minzhe Guo. “Security data mining in an ontology for vulnerability management”. In: *2009 International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing*. IEEE. 2009, pp. 597–603.

[58] Franz Baader et al. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[59] Artem Vorobiev and Nargiza Bekmamedova. “An ontology-driven approach applied to information security”. In: *Journal of Research and Practice in Information Technology* 42.1 (2010), pp. 61–76.

[60] P Salini and J Shenbagam. “Prediction and classification of web application attacks using vulnerability ontology”. In: *International Journal of Computer Applications* 116.21 (2015).

[61] Yanxiang He et al. “Ontology based cooperative intrusion detection system”. In: *IFIP International Conference on Network and Parallel Computing*. Springer. 2004, pp. 419–426.

[62] Alessandro Oltramari et al. “Building an Ontology of Cyber Security.” In: *STIDS*. Citeseer. 2014, pp. 54–61.

[63] Curtis L Maines et al. “A cyber security ontology for BPMN-security extensions”. In: *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*. IEEE. 2015, pp. 1756–1763.

[64] Miguel-Angel Sicilia et al. “What are information security ontologies useful for?” In: *Research Conference on Metadata and Semantics Research*. Springer. 2015, pp. 51–61.

[65] Salvatore Gaglio and Giuseppe Lo Re. *Advances onto the Internet of Things*. Vol. 349. Springer, 2014.

[66] Mark E Orwat, Timothy E Levin, and Cynthia E Irvine. “An ontological approach to secure MANET management”. In: *2008 Third International Conference on Availability, Reliability and Security*. IEEE. 2008, pp. 787–794.

[67] JE López De Vergara et al. “Ontologies: Giving semantics to network management models”. In: *IEEE network* 17.3 (2003), pp. 15–21.

[68] Alessandra De Paola et al. “A network ontology for computer network management”. In: *Tech. Rep. 22* (2003).

[69] Sameera Abar et al. “Exploiting domain ontologies and intelligent agents: an automated network management support paradigm”. In: *International Conference on Information Networking*. Springer. 2006, pp. 823–832.

[70] Palanivel Kodeswaran et al. “Utilizing semantic policies for managing BGP route dissemination”. In: *IEEE INFOCOM Workshops 2008*. IEEE. 2008, pp. 1–4.

[71] Cataldo Basile et al. “Ontology-based policy translation”. In: *Computational Intelligence in Security for Information Systems*. Springer, 2009, pp. 117–126.

[72] John Pinkston et al. “A target-centric ontology for intrusion detection”. In: *In proceeding of the IJCAI-03 Workshop on Ontologies and Distributed Systems. Acapulco, August 9 th*. Citeseer. 2004.

[73] Sandeep Kumar. “Classification and detection of computer intrusions”. PhD thesis. Purdue University, 1995.

[74] Cristina Abad et al. “Log correlation for intrusion detection: A proof of concept”. In: *19th Annual Computer Security Applications Conference, 2003. Proceedings.* IEEE. 2003, pp. 255–264.

[75] Nong Ye et al. “Probabilistic techniques for intrusion detection based on computer audit data”. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 31.4 (2001), pp. 266–274.

[76] Hugh E Williams and David Lane. *Web Database Applications with PHP and MySQL: Building Effective Database-Driven Web Sites*. " O’Reilly Media, Inc.", 2004.

[77] Anastasios Stasinopoulos, Christoforos Ntantogian, and Christos Xenakis. “Commix: Automating evaluation and exploitation of command injection vulnerabilities in web applications”. In: *International Journal of Information Security* 18.1 (2019), pp. 49–72.

[78] KR Srinath. “Python–the fastest growing programming language”. In: *International Research Journal of Engineering and Technology* 4.12 (2017), pp. 354–357.

[79] R Sivakumar and PV Arivoli. “Ontology visualization PROTÉGÉ tools–a review”. In: *International Journal of Advanced Information Technology (IJAIT) Vol* 1 (2011).

[80] Oscar Corcho et al. “Building legal ontologies with METHONTOLOGY and WebODE”. In: *Law and the semantic web*. Springer, 2005, pp. 142–157.

[81] Aditya Kalyanpur et al. “Debugging unsatisfiable classes in OWL ontologies”. In: *Journal of Web Semantics* 3.4 (2005), pp. 268–293.