الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

Université 20 Août 1955- Skikda
Faculté des Sciences
Département d'Informatique

جامعة 20 أوت 1955 سكيكدة
كلية العـلوم
قسم : الإعلام الآلـــي

Thèse

En vue de l'obtention du diplôme de

Doctorat de 3° cycle (LMD) en Informatique

Option : ***Computation et Cognition des Systèmes Informatiques***

Approches Collectives et Coopératives en Sécurité des Systèmes Informatiques

Présentée par :

Mohamed BELAOUED

Soutenue publiquement : 06 Juin 2016

Devant la Commission du Jury composé de :

JURY

M. Mohamed Redjim, Professeur, Université 20 Août 1955-Skikda, Président
Mme. Zizette Boufaida, Professeur, Université A.Mehri Constantine 2, Examinatrice
M. Mahmoud Boufaida, Professeur, Université A.Mehri Constantine 2, Examinateur
M. Mohamed Batouche, Professeur, Université A.Mehri Constantine 2, Examinateur
M. Bachir Boucheham, Professeur, Université 20 Août 1955-Skikda, Examinateur
M. Smaine Mazouzi, MC 'A', Université 20 Août 1955-Skikda, Directeur de thèse

Approches Collectives et Coopératives en Sécurité des Systèmes Informatiques

M. Belaoued

2016

Remerciements

Je tiens à exprimer toute ma reconnaissance à mon Directeur de thèse Dr. Smaine Mazouzi. Je le remercie de m'avoir encadré, orienté, aidé, et conseillé.

J'exprime également tous mes remerciements aux membres de jury de soutenance. M. Redjimi Mohamed, Professeur de l'université de Skikda de bien vouloir accepter de présider le jury. Mme. Boufaida Zizette, Professeur de l'université de Constantine2, M. Boufaida Mahmoud, Professeur de l'université de Constantine2, M. Batouche Mohamed, Professeur de l'université de Constantine2, M. Boucheham Bachir, Professeur de l'université de Skikda, de bien vouloir accepter d'évaluer mon travail.

J'adresse mes sincères remerciements à tous les enseignants au sein du département d'informatique de l'université de Skikda.

Je remercie mes très chers parents, qui ont toujours été là pour moi et qui sans eux je ne serais pas là aujourd'hui «Vous avez tout sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous m'avez donné un magnifique modèle de labeur et de persévérance. Je suis redevable d'une éducation dont je suis fier».

Je remercie mon frère Mourad et sa femme Sihem ainsi que ma sœur Awataf et ma sœur Sonia et son mari Mehdi, pour leur encouragement, et sans oublier mes neveux et nièces Lamis, Meria, Ahmed et Islam, vous me procurez une joie immense.

Je remercie très spécialement mes collègues et amis ; Dr.Seddari Noureddine, M. Bougeroua Salah, M. Nacer Chenniki, Fawzi, Noureddine, Abderahmen, Chawki,...Pour leur sincère amitié et confiance, et à qui je dois ma reconnaissance et mon attachement.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Résumé

Les malwares (logiciels malveillants) représentent une réelle menace pour la sécurité de nos systèmes informatiques, et avec la constante prolifération et l'évolution des techniques d'anti-détection de ces derniers, il est devenu primordial d'avoir une protection efficace contre ce genre de menaces. Malheureusement, les antivirus commerciaux sont incapables de fournir le degré requis de protection. Ceci est dû principalement au fait que ces derniers utilisent des méthodes de détection basées sur les signatures. Ces techniques sont connues pour leurs limites dans la détection des malwares inconnus, ainsi que les variantes de malwares existants. Durant ces vingt dernières années, les chercheurs en sécurité informatique ont proposé une multitude d'approches afin de remédier aux faiblesses des systèmes à base de signatures. Cependant, la majorité de ces approches focalisent sur l'amélioration du degré de précision et ignorent un facteur primordial qui est le temps de détection. En effet, être en mesure de détecter et de neutraliser une menace dans un temps court peut s'avérer vital pour la sécurité du système informatique. En plus, connaitre la nature de la menace (dans notre cas le type de malware) est également un facteur important qui va conditionner le choix des mesures à prendre. Par ailleurs, nous pensons que quelque soit son degré de précision, un outil détection local agissant d'une manière isolée sera rapidement submergé, et cela à cause du nombre colossal de malwares circulant sur le réseau internet. Dans cette thèse, nous proposons en premier lieu un système temps réel pour la détection des malwares PE (Portable Executable), en cherchant un bon compromis entre précision de détection et temps de traitement. Deuxièmement, nous proposons une nouvelle approche basée sur la méthode d'analyse des correspondances multiples (ACM) qui permet d'extraire les associations d'APIs (Application Programming Interface) utilisées par les différents types de malwares. Ces associations seront d'une grande importance pour l'identification des différents types de malwares. Enfin, nous proposons une nouvelle approche pour la détection collaborative des malwares, en utilisant les systèmes multi-agents (SMA). Notre approche offre un mécanisme de distribution et de collaboration à l'aide d'agents autonomes, permettant de faire collaborer différents outils de détection de nature hétérogènes et ayant des performances variables. Il sera également question d'envisager une méthode d'identification universelle de fichiers exécutables en utilisant une signature à base de code-opérations (Opcodes). Ainsi, la décision collective qui en résulte permet d'améliorer significativement la précision de détection.

Mots Clé : *Détection des malwares ; Méthodes statistiques ; Tests d'hypothèses ; analyse des correspondances multiples ; Systèmes Multi-Agents ; Décision Collective.*

Abstract

Malware represent a real threat to the security of our computers, and with the continued proliferation and the development of anti-detection techniques, it has become vital to have effective protection against such threats. Unfortunately, commercial antivirus software are not able to provide the required level of protection, mainly because they use signature-based detection techniques. These techniques are known for their limitations in the detection of unknown malware as well as variants of existing ones. During the past twenty years, security researchers have introduced a variety of approaches to address the weaknesses in signature-based detection systems. However, most of these approaches focus on improving the accuracy and ignore an important factor, which is the detection time. Indeed, being able to detect and neutralize a threat in a short time can be vital for the security of the system. In addition, knowing the nature of the threat (in this case the malwares type) is also an important factor that will determine the nature of the measures to be taken. Finally, we believe that whatever its degree of accuracy, a local tool working in an isolated way will be quickly overwhelmed, and this is due to the huge number of malware circulating on the Internet.

In this thesis, we propose first a real time system for detecting PE (Portable Executable) malware. In this first contribution, we tried to find a good compromise between detection accuracy and processing time. Second, we introduce a new approach based on a data anaysis method, namely the multiple correspondence analysis (MCA) that extracts the different associations of APIs (Application Programming Interfaces) used by different types of malware. These associations will be of great importance for the identification of different types of malwares.

Finaly, we propose a novel approach for collaborative malware detection using multi-agent systems (MAS). Our approach provides a cooperative mechanism with autonomous agents that will allow the collaboration of different heterogeneous malware detection tools. We will also discuss the nececisty to have a universal identification method for executable files using an Opcode-based signature. Thus, the collective decision that results can significantly improve the detection accuracy.

Keywords : *Malware Detection ; Statistical Methods ; Hypotheses Testing ; Mutliple Correspondance Analysis ; Multi-Agent Systems ; Collective Decision.*

ملخص

البرمجيات الخبيثة تمثل تهديدا حقيقيا لأمن أنظمتنا الحاسوبية. ومع استمرار انتشار وتطوير تقنيات الكشف المضادة لهذه البرمجيات، فقد أصبح من الضروري توفير حماية فعالة ضد هذا النوع من التهديدات. للأسف، التطبيقات التجارية المتخصصة في مكافحة الفيروسات ليست مؤهلة لتوفير المستوى المطلوب من الحماية وهذا راجع إلى اعتمادها على الأساليب التحليلية القائمة على التوقيعات. هذه التقنيات معروفة بقصورها في الكشف عن البرامج الضارة غير المعروفة و متغيرات البرمجيات الخبيثة الموجودة. خلال العقدين الماضيين، اقترح الباحثون في الأمن المعلوماتي مجموعة متنوعة من مناهج اكتشاف البرامج الضارة وذلك لمعالجة نقاط الضعف في الأنظمة القائمة على التوقيعات .ومع ذلك، فإن معظم هذه المناهج تركز على تحسين درجة الدقة وتجاهل عامل رئيسي ألا وهو زمن الكشف .في الواقع، في مثل هذه الأنظمة، القدرة على كشف وتحييد نوع التهديد في وقت قصير يمكن أن تكون حيوية لأمن النظام .وبالإضافة إلى ذلك، معرفة طبيعة التهديد (في هذه الحالة نوع البر نامج الخبيث) هو أيضا أحد العوامل الهامة التي ستحدد طبيعة الإجراءات اللازم اتخاذها . وأخيرا، فإننا نعتقد أن أيا كانت درجته من الدقة، نظام الكشف الذي يعمل بطريقة منعزلة سيتم اجتيازه بسرعة وهذا نظرا لوجود عدد كبير من البرمجيات الخبيثة المنتشرة على شبكة الإنترنت، و كذلك تلك التي يتم اكتشافها يوميا.

في هذه الأطروحة، نقترح . في خطوة أولى . نظام بخاصية الوقت حقيقي للكشف عن البرمجيات الضارة من نوع (PE) (محمول قابل للتنفيذ). في إطار هذه المساهمة الأولى، حاولنا أن نجد حلا وسطا بين دقة الكشف ووقت المعالجة.

ثانيا، نقترح نهجا جديدا يقوم على أسلوب تحليل الموافقات المتعددة (MCA) الذي يستخرج مختلف الروابط الموجودة بين واجهات برمجة التطبيقات (API) المستخدمة من قبل مختلف أنواع من البرمجيات الخبيثة.

وأخيرا، نقترح منهج كشف تعاوني عن البرمجيات الضارة وذلك باستخدام الأنظمة المتعددة العملاء (MAS). منهجنا يوفر آلية توزيع استنادا على العملاء المستقلين، هذه الأخيرة تتيح الفرصة لأدوات الكشف غير المتجانسة من أجل التعاون، وذلك بتفاوت في الأداء من أداة إلى أخرى ، ما يسمح أساسا لبلوغ قرار تعاوني وتوافقي حول وجود تهديد ذو نوع خبيث من عدمه. ينبغي أن يساهم هذا إلى حد كبير في تحسين دقة الكشف.

الكلمات المفتاحية: الكشف عن فيروسات الكمبيوتر؛ الأساليب الإحصائية، اختبار الفرضيات، الأنظمة المتعددة العملاء، قرار تعاوني وتوافقي.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Sommaire

1.1 Contexte

À notre ère moderne, les systèmes informatiques peuplent notre quotidien et interviennent dans toutes nos activités, des plus domestiques aux plus professionnelles. Cependant, et avec la croissance fulgurante et la banalisation de l'accès au réseau internet, ces systèmes sont devenus sujets à des cyber-attaques qui causent chaque année des dégâts financiers estimés à plusieurs centaines de milliards de dollars [1]. Ces pertes sont dues généralement aux coûts de réparation et de restauration des systèmes endommagés, mais aussi à la baisse de productivité des employés, ainsi que les préjudices occasionnés par les retards dans les services qui doivent être accomplis auprès des clients [Erbschloe, 2004, Malin et al., 2011]. Certaines pertes, sont difficiles à estimer, notamment l'image de marque de l'entreprise et la confiance des clients. Le terme cyber-attaque est utilisé pour désigner toute action menée par des Hackers (pirates informatiques, ou cyber criminels) dont le but est de modifier, détruire, ou subtiliser des données, ou de perturber le bon fonctionnement d'un système informatique ou d'un réseau de communication [Dua and Du, 2011, Jensen, 2002]. Une attaque peut être aussi définie comme étant : *une faute d'interaction externe malveillante visant à violer volontairement une ou plusieurs propriétés de sécurité* (voir le chapitre 2, section 2.2) [Sang, 2012].

Les codes malveillants, ou plus communément connus par leur appellation anglo-saxonne *'malwares'* (pour Malcious Software) sont l'arme de choix des Hackers pour mener leurs attaques [Jang-Jaccard and Nepal, 2014, Erbschloe, 2004]. *Un malware est un programme*

1. Net Losses : Estimating the Global Cost of Cybercrime, McAfee Labs (2014) : `http://www.mcafee.com/us/resources/reports/rp-economic-impact-cybercrime2.pdf`

informatique qui a été élaboré afin de mener une intrusion ou provoquer des dommages à un système informatique [Salehi et al., 2014]. Il existe une grande variété de malwares, tels que les virus, les vers, les chevaux de trois ...etc. Plusieurs classifications, selon différents critères existent dans la littérature, et dont nous aurons à détailler dans un chapitre qui leur est consacré.

Les origines des malwares remontent aux années 1940 avec les travaux du scientifique hongrois John Von Neumann sur les automates autoreproducteurs [Szor, 2005]. Von Neumann évoquait pour la première fois la possibilité de conférer aux programmes informatiques la capacité de se répliquer d'une manière autonome. Plus de quarante ans plus tard, en 1987 plus précisément, le chercheur américain Fred Cohen a défini formellement le terme de 'virus informatique', comme étant *un programme qui peut en infecter d'autres en modifiant leur contenu* [Cohen, 1987].

Ces dernières années nous avons assisté à une inquiétante prolifération des malwares. En effet, selon les statistiques fournies par l'institut indépendant spécialisé en sécurité informatique AvTest [2], le nombre de nouveaux codes malveillants n'a cessé de s'accroitre durant les cinq dernières années. Par exemple, on a recensé pas moins de deux millions de nouveaux malwares en 2011, et pas moins de 8 millions de malwares en 2013, et ce nombre est passé à plus de 140 millions en 2014 et en 2015. L'autre fait inquiétant concernant cette menace est que les concepteurs de malwares ne cessent de surpasser les mesures de protection, en créant à chaque fois des codes qui sont de plus en plus difficiles à détecter, et ce en introduisant des techniques d'obscurcissement de plus en plus efficaces. Devant cette prolifération et évolution des malwares, et la vitalité des activités électroniques via l'Internet, il est primordial aujourd'hui, plus que jamais, de développer des systèmes qui permettront de protéger efficacement nos applications informatiques contre les cybercriminels. La proposition de nouveaux systèmes de protection contre les malwares, passe inévitablement par la proposition de nouvelles approches de détection de codes malveillants. Une multitude d'approches visant la performance de détection à travers l'analyse du code lui-même sont proposées dans la littérature [Zhou et al., 2010]. Ces approches visent à augmenter le degré de différentiation entre un code sain et un code malveillant, et ainsi, proposer des classifieurs performants pour la détection de malwares. Par ailleurs, et étant donnée la forte connectivité des systèmes informatiques, via l'Internet, des approches collectives et coopératives de détection peuvent être proposées. L'objectif de telles approches est de renforcer le degré de certitude de détection d'un code exécutable donné, calculé au niveau d'une entité donnée, en échangeant les informations de détection avec d'autres entités ayant procédé à la détection du même code. Les difficultés qui se trouvent en face de telles approches sont nombreuses. Il s'agit de l'étendu

2. AvTest : `www.av-test.org/fr/`

du réseau d'Internet d'une part, et de l'uniformisation des paramètres de détection et de nommage d'une autre part, pour ne citer que deux difficultés, auxquelles nous nous sommes, entre autres, intéressés dans le cadre de cette thèse.

1.2 Problématique

En considérant la littérature du domaine, on peut facilement observer que toute technique de détection de malware a ses propres avantages et inconvénients. Nous pouvons donc conclure qu'il est impossible d'avoir une protection qui soit complément efficace et performante. Cependant, et en se basant sur nos analyses, nous pouvons avancer que les méthodes basées sur l'apprentissage automatique semblent être les plus adéquates pour offrir un bon degré de protection, à condition de pouvoir trouver un bon rapport entre précision et rapidité. En effet, ces deux paramètres (la précision et le temps de détection) dépendent étroitement de la nature ainsi que du nombres des attributs utilisés, la façon dont ils sont extraits, et l'algorithme de classification utilisé.

Il est également souhaitable d'associer les sous-ensembles d'attributs caractérisant les différents malwares en leurs différentes classes respectives. Il sera judicieux de pouvoir catégoriser les codes malveillants en les groupant dans des classes, où chacune correspond à un type donné de malwares. Mais aussi en utilisant un sous-ensemble d'attributs caractéristiques, identifiant le groupe en question.

Par ailleurs, quand on procède à coupler des sous-systèmes de détection pour les faire coopérer, maintes problèmes sont à résoudre. En premier lieu, et étant donné le polymorphisme élevé qui caractérise les codes malwares, il faut mettre en place un système d'identification universel, et de nommage pour les codes malveillants, nécessaire pour pouvoir faire communiquer des entités distantes dans le but de coopérer à détecter un code malveillant. En second lieu, il faut mettre en place des mécanismes cohérents d'intégration de l'information et de la connaissance, qui proviennent des différents nœuds, et qui représentent des décisions qui peuvent être inconsistantes.

1.3 Contribution

Dans cette thèse, nous avons essayé de d'apporter des contributions liées aux problématiques citées précédemment. Notre première contribution est motivée par le fait que les techniques basées sur l'apprentissage automatique ont le potentiel de nous permettre d'effectuer une analyse en temps réel des codes malveillants avec un taux de précision assez important. De ce fait, notre première contribution consiste en :

- L'utilisation d'une méthode statique pour l'extraction des attributs qui est caractérisée par sa grande rapidité.
- L'utilisation d'une méthode simple pour la représentation des attributs.
- L'utilisation d'une méthode basée sur le test d'hypothèses pour la sélection des attributs, qui est à la fois rapide et efficace.
- Évaluer plusieurs algorithmes de classification afin de pouvoir sélectionner le plus performant d'entre eux, en considérant les attributs sélectionnés.

Deuxièmement, et afin de remédier au problème lié à la catégorisation multi-classes des malwares, nous avons proposé une nouvelle approche basée sur une méthode d'analyse de données, à savoir, l'analyse des correspondances multiples (ACM). Celle-ci nous a permis d'identifier les différentes associations d'APIs (Application Programming Interfaces) utilisées par les différents types de malwares. En se basant sur ces associations, nous pourrons distinguer les différents types de malwares.

Troisièmement, nous proposons un mécanisme de décision consensuelle, distribué pour la détection des malwares qui sera basé sur un ensemble d'agents, implémentés chacun au niveau d'un nœud du réseau. La motivation derrière cette dernière, est de pouvoir améliorer l'efficacité des outils d'analyse et de détection des malwares qui différent d'une machine à une autre, en les faisant collaborer, tout en assurant un système de nommage et d'identification universel des codes exécutables.

1.4 Organisation du manuscrit

Le long de ce manuscrit, nous allons revenir aux problématiques considérées, et aux différentes contributions apportées, tout en essayant d'insérer notre travail dans la littérature de ce domaine. De ce fait, nous avons voulu organiser notre manuscrit comme suit :

Le second chapitre est un survol des principaux aspects liés à la sécurité des systèmes informatiques d'une façon générale, en mettant, cependant, l'accent sur les menaces liées aux réseaux et aux systèmes d'exploitation Microsoft Windows, ainsi que sur les méthodes de protection existantes.

Le troisième chapitre sera consacré au domaine de la virologie informatique. Dans ce chapitre, nous allons nous étaler sur les différents concepts liés aux malwares, et plus précisément aux malwares PE (Portable Executable) élaborés pour les systèmes d'exploitation Windows.

Le chapitre quatre, sera divisé en deux parties, la première partie est consacrée à notre approche statistique pour la détection en temps réel des malwares PE. La seconde partie quant

à elle, sera consacrée à notre approche basée sur l'ACM pour l'extraction des associations d'APIs pour la catégorisation multi-classes des malwares. Le chapitre cinq, sera consacré à notre approche collective et distribuée pour la détection de malwares, où nous présenterons notre système orienté SMA (systèmes multi-agnets) pour la détection collaborative des malwares. Une conclusion générale, viendra récapituler ce qui a été réalisé le long de cette thèse, et souligner nos travaux futurs, ainsi que mettre le doigt sur certaines perspectives relatives à la sécurité des applications informatiques.

Chapitre 2

Sécurité des Systèmes Informatiques

Sommaire

2.1 Introduction

Avec l'étendue et la généralisation des systèmes informatiques communicants, la sécurité informatique, est devenue primordiale, et d'une priorité absolue pour la bonne exploitation de ces systèmes. Le domaine de la sécurité informatique traite les différents moyens mis en place pour assurer la sécurité des systèmes informatiques. Il s'agit entre autre, des moyens matériels et humains, ainsi que des procédures organisationnelles qui leurs sont inhérentes. J.F Pillou l'a résumé en avançant que *la sécurité informatique consiste à garantir que les ressources matérielles ou logicielles d'une organisation, sont uniquement utilisées dans le cadre qui leur est prévu* [Pilou, 2006]. De nos jours, tout système informatique, allant d'un PC ou un Smartphone, jusqu'aux grandes installations informatiques, composées de mainframes, doit être rigoureusement sécurisé, et ce, en prévoyant la politique et les différents

outils de sécurité qui leur sont nécessaires. Au-delà des dégâts financiers, qui s'élèvent à des dizaines, voir des centaines de milliards de dollars par an, l'information privée est d'une importance capitale, dont le vol consiste en un préjudice non estimable pour les personnes, les organisations, et aussi pour les États.

Créer un système informatique, au départ complètement sécurisé, relève de l'utopie. En effet, les outils de développement des systèmes informatiques, langages, bibliothèques, ...etc, génèrent souvent des vulnérabilités, à l'insu des développeurs qui les utilisent. Aussi, l'humain est lui-même, dans sa nature vulnérable, et tend à simplifier les coûts en passant à côté des consignes de sécurité au cours du développement des applications informatiques.

La sécurité informatique s'applique aux différents niveaux d'un système informatique. En allant du haut vers le bas, elle concerne, les réseaux, les applications, le système d'exploitation, et les ressources matériels [Kern et al., 2007]. Au niveau le plus haut, il est demandé que les informations qui circulent via les réseaux, passant d'un nœud à un autre, soient protégées contre tout accès illégal, que ça soit de déni, d'altération, ou de vol. Au second et troisième niveaux, l'objectif est d'assurer que les applications et les bibliothèques systèmes développées et utilisées soient dépourvues de failles, et de vulnérabilités permettant aux hackers, en les exploitant, de nuire aux systèmes sur lesquelles elles tournent. Le plus dangereux, est que ces failles permettent carrément l'intrusion à l'intérieur des systèmes, pour voler de l'information, ou plus gravement l'installation de codes malveillants.

La sécurité du matériel prévoit le rapprochement physique aux systèmes informatiques. Cette dernière, concerne donc toutes les mesures de sécurité laissant les installations informatiques, à l'abri de toute personne, sauf celles qui sont autorisées, et ce via un contrôle rigoureux de la circulation et de l'accès au sein des locaux où le matériel informatique est installé.

Dans le reste de ce chapitre, nous commencerons par présenter les différentes propriétés relatives à la sécurité de l'information, ensuite, nous proposerons une classification des attaques informatiques selon différents aspects, et enfin, nous aborderons les différents moyens et techniques de protection contre les attaques informatiques.

2.2 Quelques propriétés en sécurité informatique

Le Parkerian Hexad qui a été proposé par Don B.Parker en 1998 [Parker, 1998] est un ensemble de six éléments fondamentaux, qui doivent être protégés par les mesures de sécurité informatique. Ces six éléments sont comme suit :

– La confidentialité : définit les restrictions relatives aux accès accordés ou refusés à telle ou telle personne. Cela revient à garder secrète, une information ou une communication

quelconque.

- La possession ou le contrôle : traite le problème de perte du contrôle sur une information sans pour autant affecter la confidentialité.
- L'intégrité : représente le fait que l'information reste correcte, or toute modification non-autorisée de celle-ci représente une violation de l'intégrité.
- L'authenticité : se réfère à la véracité de la revendication de l'origine ou de l'auteur de l'information, donc, ceci se résume en l'aptitude à pouvoir vérifier l'identité d'une personne, ou d'une source.
- La disponibilité : signifie avoir accès en temps opportun à l'information, cela relève de la performance de la sécurité d'un système [Kern et al., 2007].
- L'utilité : signifie s'assurer de ne pas posséder des informations inutiles, à savoir, une information dupliquée ou indéchiffrable, telle une donnée cryptée dont le code de déchiffrement a été oublié.

D'autres caractéristiques peuvent venir élargir la liste précédente [Kern et al., 2007], telles que :

- La responsabilité : l'objectif de la responsabilité est de s'assurer d'être en mesure de déterminer l'identité de l'attaquant ou bien le responsable dans le cas où un problème ou une transaction erronée est identifiée.
- La non-répudiation : l'objectif de la non-répudiation est d'assurer le caractère indéniable d'une transaction par l'une des parties impliquées.

2.3 Classification des attaques informatiques

Il existe dans la littérature plusieurs travaux proposant des taxonomies pour la classification des attaques informatiques. Abbott et al. [Abbott et al., 1976], et Bisbey et Hollingworth [Bisbey and Hollingworth, 1978] ont proposé en 1976, et en 1978 respectivement deux classifications assez similaires vu que toutes les deux sont plus basées sur le concept de vulnérabilités que celui des attaques. Celles-ci ont été d'une grande importance vu qu'elles ont permis l'avènement de nouvelles taxonomies telles que celles proposée par S.Kumar [Kumar, 1995] en 1995 et celle de Bishop and Bailey en 1996 [Bishop, 1999], qui sont venues enrichir les deux classifications précédentes, en introduisant la notion d'attaque informatique.

K.Boudaoud [Boudaoud, 2002] a proposée en 2001 une classification selon quatre critères tels que la nature de l'attaquant (utilisateur interne ou externe), la propriété du système mise en cause (la confidentialité, l'intégrité, etc.), ainsi qu'une classification scindant les attaques informatiques en deux types qui sont les attaques réseaux et les attaques systèmes.

Enfin, la taxonomie proposée par Hansman et Hunt [Hansman and Hunt, 2005] en 2005

suggère que les attaques informatiques peuvent être réparties suivant quatre dimensions qui sont : le vecteur d'attaque, la nature de la cible, les vulnérabilités exploitées, et enfin la charge (Payload). Dans cette thèse, nous allons classifier les attaques informatiques en s'inspirant de la classification proposée par Hansman et Hunt [Hansman and Hunt, 2005], comme l'illustre la figure 2.1 :

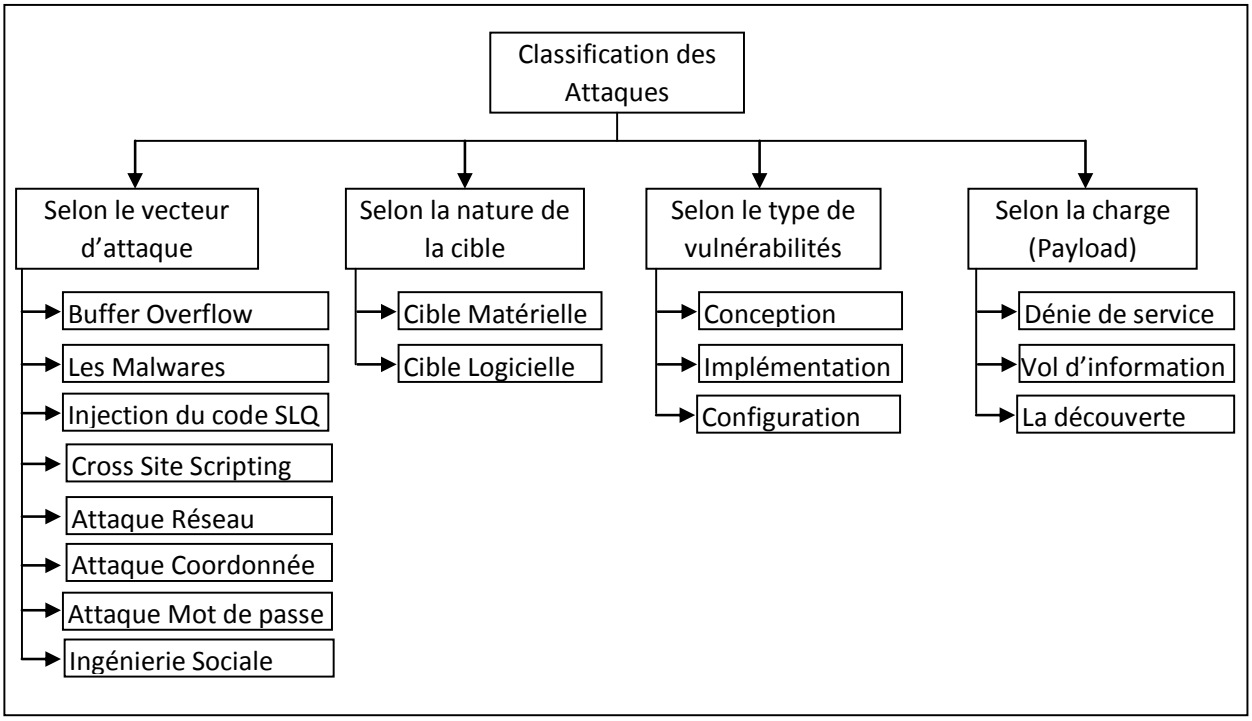

FIGURE 2.1 – Classification des attaques informatiques

Notre choix est motivé par le fait que cette classification est la plus récente ainsi que la plus pertinente des taxonomies ayant été proposées à ce jour. En outre, nous pensons qu'elle est la plus adéquate pour le domaine de la détection de codes malveillants. Selon cette décomposition, les attaques informatiques sont classées comme suit :

2.3.1 Classification selon le vecteur d'attaque

Cette dimension regroupe l'ensemble des moyens utilisés par l'attaquant afin d'atteindre sa cible [Hansman and Hunt, 2005]. Les différents vecteurs d'attaques sont :

Le débordement de capacité du tampon (Buffer Overflow)

Le tampon (buffer en anglais) est une zone de stockage qui peut contenir une quantité finie d'informations. Un dépassement de capacité du tampon 'buffer overflow' se produit lors-

qu'un programme essaie de stocker dans le tampon des données dont la taille excède la taille du tampon. Lorsque les données débordent du tampon, les données supplémentaires peuvent déborder dans des emplacements de mémoire adjacents, et ainsi corrompre les données valides et, éventuellement, modifier le chemin d'exécution et les instructions. La capacité d'exploiter un débordement de tampon permet éventuellement d'injecter du code dans le chemin d'exécution. Ce code ajouté pourrait permettre l'accès au système à distance car il s'exécute en zone locale, donnant un accès non autorisé non seulement aux hackers, mais aussi aux malwares [Chien and Ször, 2002]. Ce type d'attaques est souvent utilisé par les malwares, l'exemple le plus connu reste le ver Morris (voir chapitre 3).

Les Malwares

Exécuter un code en zone locale d'une machine permet la possession de son contrôle, ou au moins avoir tous les droits de l'utilisateur dont sa session était utilisée lors de l'exécution du code. Si un attaquant arrive à exécuter un code dans la zone locale, soit en exploitant une vulnérabilité ou par ingénierie sociale, ceci est le pire qu'une machine victime peut subir comme préjudice. On appelle dans ce cas le code exécuté : un malware.

Les malwares sont généralement repartis en deux catégories représentant les codes simples et les codes autoreproducteurs [Filiol, 2009]. La première catégorie représente les malwares qui sont résidants dans la machine victime, et qui s'exécutent suite à un événement (une action, une date précise, etc.). Quant à la seconde catégorie, elle représente les malware ayant une certaine autonomie et pouvant se propager dans la même machine (duplication) ou bien d'une machine à une autre à l'aide d'un moyen de communication tel qu'un réseau, ou par courriel [Filiol, 2009]. Un malware passe généralement par trois phases dans son cycle de vie, qui sont l'installation, la communication, et l'exécution [Jacob et al., 2009]. La phase d'installation reste la phase la plus importante, et qui est généralement effectuée en exploitant une vulnérabilité dans la machine cible. Le Buffer Overflow, par exemple, reste le moyen d'excellence pour l'installation des malwares. Dans le chapitre suivant nous allons aborder en détails les différents concepts liés aux malwares.

Injection de code SQL

L'attaque par injection de code SQL cible les applications Web, et celle-ci permet à un attaquant d'obtenir un accès à leurs bases de données et d'obtenir des informations sensibles sur les utilisateurs. Les violations de sécurité qui en résultent peuvent inclure le vol d'identité, la perte d'informations confidentielles, et la fraude [Halfond et al., 2006]. Le site du géant de l'audiovisuel Sony (Sonypictures.com) était victime en 2011 d'une attaque de type injection

de code SQL, qui en résulta la subtilisation de plus d'un millions de comptes personnels [1]. Dans ce type d'attaques, l'attaquant essaie d'injecter du code SQL dans une requête de type GET ou POST [Ballmann, 2015]. Prenons l'exemple d'une application web qui est vulnérable à la vérification du type d'informations saisies et où un attaquant veut se connecter à celle-ci sans disposer du mot de passe, celui-ci pourrait utiliser les informations d'authentification suivantes :

Nom d'utilisateur : ALI';--
Mot de passe : n'importe lequel (ex : ABCD)
Dans ce cas, le script SQL suivant serait exécuté :

```
SELECT uid
FROM Utilisateurs
WHERE nom = 'ALI'; -- ' AND password ='ABCD';
```

La partie `-- ' AND password ='ABCD';` de la requête ne sera pas exécutée, vu qu'elle est précédée par le caractère `--` qui marque le début d'un commentaire en SQL. Par conséquent, l'attaquant peut se connecter avec le nom d'utilisateur ALI et n'importe quel mot de passe.

Cross-site Scripting (XSS)

L'attaque par Cross-site Scripting (XSS) cible les applications web, et consiste à forcer le navigateur web de la victime à exécuter un script (généralement de type JavaScript) dans le but, par exemple, de subtiliser les informations de connexion (stockées dans des cookies de session) [Ballmann, 2015]. Plusieurs facteurs contribuent à la prévalence de la vulnérabilité à une attaque de type XSS. Cependant, cette attaque touche généralement les applications Web qui permettent d'afficher une entrée non fiable (sans vérifier son encodage). Dans ce cas précis, un attaquant peut injecter son script malicieux dans la page Web, via un champ de recherche par exemple, et une fois que le code est exécuté, l'attaquant aura accès à l'ensemble des informations échangées entre le serveur et la page web de l'utilisateur [Wassermann and Su, 2008].

Les attaques réseaux

Le but principal de ce type d'attaques et d'empêcher les utilisateurs d'utiliser une connexion réseau, de rendre indisponible une machine ou un service et de surveiller le trafic réseau dans le but de l'analyser et d'en récupérer des informations pertinentes [Boudaoud, 2002]. Comme

1. Sony investigating another hack, BBC news (2011) : `http://www.bbc.com/news/business-13636704`

exemple d'attaques réseaux on peut citer :

- L'usurpation d'adresses IP (IP Spoofing) : C'est une technique de détournement dans laquelle le Hacker dissimule son identité en se faisant passer pour un hôte approuvé en envoyant au serveur des paquets dont l'entête a été modifié en y incluant une adresse IP source d'un hôte légitime [Boudaoud, 2002]. Le but de cette attaque est de détourner un site Web, obtenir accès à un réseau, et d'obtenir des droits d'accès élevés [Boudaoud, 2002, Ballmann, 2015].

- L'inondation ICMP (ICMP Flooding) : Le but de cette attaque est d'inonder le réseau en envoyant à plusieurs reprises, et sans attendre une réponse, un grand nombre de paquets ICMP (Internet Control Message Protocol) "Echo Request" avec comme adresse IP de destination l'adresse de diffusion (broadcast) du réseau cible [Boudaoud, 2002]. A la réception des paquets ICMP, le routeur les diffuse sur toutes les machines du réseau ce qui va entrainer sa congestion ou sa mise hors service. Un cas particulier de l'ICMP Flooding est l'attaque Smurf, celle-ci consiste à remplacer l'adresse source par l'adresse de la machine cible, qui recevra toutes les réponses "ICMP Reply" des hôtes atteints par les paquets ICMP "Echo Request" [Gadelrab, 2008].

- L'inondation TCP-SYN (TCP-SYN Flooding) : Son principe consiste à envoyer un grand nombre de requêtes TCP-SYN à un service (souvent HTTP) sur la machine cible avec une adresse source erronée (inexistante). Cela va engendrer la création de connexions semi-ouvertes et l'envoie de réponses sous forme de requêtes SYN-ACK. Celles-ci ne seront jamais reçues et le message ACK ne sera pas généré, ce qui causera la saturation de la file d'attente des demandes de connexions, et ainsi toutes les requêtes envoyées au port TCP cible seront systématiquement ignorées, ce qui rendra le service fourni sur ce port indisponible [Boudaoud, 2002, Gadelrab, 2008, Zhou et al., 2010].

- Les renifleurs (Sniffers) : Un sniffer est un programme qui a été conçu dans le but d'observer le trafic au niveau d'un réseau en analysant les paquets qui y sont véhiculés dans le but d'avoir des informations sur la performance du réseau et de détecter d'éventuels problèmes qui peuvent survenir. Cependant, un renifleur peut être utilisé de façon illégitime en l'utilisant pour obtenir des informations pertinentes tels que les logins et les mots de passe, les adresses IP des machines, ainsi que leur rôle (Client, Serveur de fichier, serveur Web, etc), qui seront utilisées afin de préparer d'autres attaques [Boudaoud, 2002].

Les attaques coordonnées

Les attaques coordonnées consistent à utiliser un très grand nombre d'hôtes (appelés zombies) qui ont été compromis au préalable à l'aide de codes malveillants, ce qui les rends extrêmement dangereuses et très difficiles à détecter [Zhou et al., 2010]. La DDoS (pour Distibuted Denial of Service, en français dénie de service distribué), est une attaque coordonnée très redoutée car elle cause des dégâts considérables. Dans une DDoS, l'attaquant contrôle le processus d'attaque à distance des machines zombies par le biais de canaux IRC (Internet Relay Chat, en français, discussion relayée par Internet) ou le réseau pair-à-pair (peer-to-peer). Un tel réseau d'ordinateurs infectés est appelé un 'Botnet' [Jang-Jaccard and Nepal, 2014], et avec un assez grand nombre de ces hôtes zombies, même les services des sites les plus importants et les mieux dotés de ressources peuvent être perturbés [Gadelrab, 2008, Zhou et al., 2010]. Le 31 décembre 2015 le site du groupe BBC (Britsh Broadcasting Corporation) a été la cible de ce qu'on a qualifié comme étant 'une attaque DDoS record' avec une bande passante cumulée de plus 600 Gb/s, et qui a causé la mise hors service du site www.bbc.com pendant plusieurs heures [2] [3].

Attaques de mots de passes

Ce type d'attaques a pour but de deviner le mot de passe utilisé pour s'authentifier à un service ou à une application donnée. Il existe deux grandes familles d'attaques de type découverte de mot de passe qui sont l'attaque par force brute, et l'attaque par dictionnaire [Stamp, 2011].

- L'attaque par force brute : Aussi connue sous le nom d'attaque exhaustive. Elle consiste à tester toutes les combinaisons possibles du mot de passe. Elle ne peut être envisagée que si le mot de passe est assez court, permettant de tester toutes les combinaisons possibles.

- L'attaque par dictionnaire : se type d'attaque se base sur l'utilisation d'un dictionnaire de mots passes, celui-ci regroupe, par exemple, les mots de passes par défaut des applications, les mots de passes couramment utilisés, etc. Ce type de méthodes est nettement plus rapide que la méthode par force brute, et ne met aucune restriction sur la longueur du mot de passe.

Il existe d'autres moyens qui permettent d'obtenir les logins ainsi que les mots de passe des utilisateurs. Par exemple, l'attaquant peut utiliser un malware de type Keylogger (voir

2. Mirror (2016) :`http://www.mirror.co.uk/news/uk-news/anti-isis-hackers-claim-carried-7105317`.
3. Le Monde Informatique (2016) : `http://www.lemondeinformatique.fr/actualites/lire-l-attaque-ddos-record-contre-la-bbc-atteindrait-602-gb-s-63529.html`

chapitre 3), ou bien en se basant sur le facteur humain comme mode d'attaque, ce genre de technique est appelée l'ingénierie sociale et qui est discutée ci-dessous.

L'ingénierie sociale

Contrairement aux attaques mentionnées précédemment, qui peuvent être évitées via les mises à jour et les bulletins de sécurité qui permettent de combler une vulnérabilité, l'attaque par ingénierie sociale exploite la vulnérabilité humaine [LeDoux and Lakhotia, 2015]. En effet, le facteur humain reste sans aucun doute le maillon faible de la sécurité des systèmes informatiques [Mitnick and Simon, 2004]. Les attaques par l'ingénierie sociale se divisent en deux catégories qui sont les attaques classiques et les attaques automatisées [Aycock, 2006].

- Les attaques classiques : le cas le plus répondu dans ce genre d'attaques est l'usurpation d'identité, ou le Hacker va essayer de leurrer la victime en prétendant être quelqu'un d'autre et essayer d'obtenir des informations sensibles telles que des logins et des mots de passes.

- Les attaques automatisées : Ce type d'attaque est généralement menée par le biais de malwares, par exemple un malware peut essayer d'obtenir le login et le mot de passe d'un utilisateur en affichant des fenêtres pop-up lui demandant de réintroduire ses informations de connexion, ou celui-ci va convaincre l'utilisateur de cliquer sur un lien qui va le conduire au site web de l'attaquant. En outre, un malware peut être dissimulé dans un courriel, sous forme d'un attachement comme ce fut le cas avec le célèbre malware : Melissa (voir chapitre 3).

2.3.2 Classification selon la nature de la cible

Cette deuxième dimension définit la ou les cibles d'une attaque. Cela implique l'ensemble des services ciblés ainsi que le type du système d'exploitation. En outre, les pirates informatiques peuvent cibler une configuration matérielle bien précise. De ce fait, les cibles d'une attaque peuvent être divisées en deux catégories qui sont les cibles logicielles et les cibles matérielles, tout en sachent qu'une attaque peut cibler les deux en même temps.

Les cibles matérielles

Les cibles matérielles peuvent être divisées en trois différentes catégories qui sont la machine (CPU, Disques durs, etc.), les équipements réseaux (Switches, Hubws, routeurs, etc.), et les

périphériques (claviers, moniteurs, etc.) [Hansman and Hunt, 2005]. Une attaque peut cibler une ou plusieurs de ces catégories.

Les cibles logicielles

Les cibles logicielles englobent trois différentes catégories qui sont les systèmes d'exploitations, les applications, et les protocoles réseaux [Hansman and Hunt, 2005] :

- Les systèmes d'exploitation (SE) : généralement toute attaque est destinée à un SE cible. Si l'on prend des attaques par malwares, les systèmes d'exploitation Microsoft Windows restent de loin les plus ciblés par ce genre de menaces, cela est dû entre autre au fait que les SE Windows (toutes versions confondues) sont les plus répondus dans le monde [4].

- Les applications : les attaques peuvent aussi être conçues pour un type particulier d'applications, tels que les serveurs (Web, Base de données, e-mail, etc.) ou bien les applications utilisateur (Les utilitaires, les antivirus, etc.).

- Les protocoles réseaux : un attaquant peut aussi cibler un protocole réseau spécifique tels que le TCP, ou bien l'ICMP afin de rechercher ou d'exploiter des vulnérabilités qui lui sont associées.

2.3.3 Classification selon le type de vulnérabilités exploitées

Une vulnérabilité est une faute accidentelle ou intentionnelle (avec ou sans volonté de nuire) dans la spécification des besoins, la spécification fonctionnelle, la conception, l'implémentation, la configuration du système, ou dans la façon selon laquelle il est utilisé [Sang, 2012]. J.Howard [Howard, 1997] a proposé une classification des attaques informatiques en trois classes selon le type de vulnérabilités exploitées, et qui sont les vulnérabilités de conception, les vulnérabilités des configurations et les vulnérabilités d'implémentation.

- Vulnérabilités dans la conception : elles sont causées par des erreurs non-intentionnelles lors de la conception d'une application. C'est le cas par exemple pour les applications qui ne vérifient pas l'encodage des caractères saisis par l'utilisateur. Dans ce genre de cas, cela pourrait être utilisé pour mener une attaque de type XSS.

4. OS Platform Statistics, W3School (2016) :`http://www.w3schools.com/browsers/browsers_os.asp`

– Vulnérabilités dans l'implémentation : Dans le cas où le système serait bien conçu, mais son implémentation contient des bugs, cela peut compromettre sa sécurité. Un exemple de vulnérabilité d'implémentation assez récente qui a été découverte vers le mois de juillet 2015 dans le logiciel Adobe Flash, et qui a été publiée dans l'annuaire CVE [5] (Common Vulnerabilities and Exposures) sous le code CVE-2015-5122. Cette vulnérabilité pourrait être exploitée afin de mener une attaque de type Buffer Overflow.

– Vulnérabilités dans la configuration : la configuration d'un système et aussi importante que sa conception et son implémentation [Howard, 1997]. On a beau créer un système bien conçu et bien implémenté, cela ne suffira pas s'il est mal configuré. Un exemple simple de vulnérabilité de configuration est le fait de laisser un nombre important de ports ouverts. Un autre exemple très répondu concerne les comptes par défaut qui sont fournis dans bon nombres de logiciels et de serveurs. Ces derniers ont souvent comme nom d'utilisateur 'Administrateur' et comme mot de passe 'password' ou '1234'. Ces comptes peuvent être utilisés pour accéder à distance à la machine, de ce fait ils doivent impérativement être supprimés.

2.3.4 Classification selon la charge (Payload)

La quatrième dimension représente les attaques ayant une charge (Payload en anglais). La charge est l'objectif derrière le lancement de l'attaque. Si l'on prend l'exemple des malwares ceux-ci peuvent avoir ou ne pas avoir de charge, et dans le cas ou le malware à une charge elle diffère d'une catégorie de malware à une autre. Par exemple un malware de type virus, peut avoir comme charge la modification ou la suppression de fichiers. Le vol d'information est une conséquence d'un bon nombre d'attaques, à savoir, les attaques par ingénierie sociale, les Sniffers, ainsi que certains types de malwares tels que les Keyloggers. Le déni de service (DoS, pour Denial of Service) conduit généralement à une perte totale ou de la dégradation des services en consommant de la bande passante du réseau de la victime ou en surchargeant les ressources informatiques de son hôte. Le déni de service est la conséquence de certaines attaques réseaux tels que l'ICMP Flooding, SYN Flooding, ainsi que l'attaque de type DDoS. La découverte peut également être l'un des objectifs d'une attaque. Celle-ci se résume dans le fait d'utiliser les informations collectées sur une cible (par le biais, par exemple, d'un Sniffer ou d'un malware) afin de mener d'autres attaques.

5. Common Vulnerabilities and Exposures :`https://cve.mitre.org/`

2.3.5 Synthèse

Selon cette décomposition, une attaque doit impérativement avoir la première dimension. Cependant, toutes, une partie, ou aucune des autres dimensions pourraient ne pas exister [Hansman and Hunt, 2005]. En outre, une attaque peut avoir un ou plusieurs vecteurs d'attaques, ainsi qu'une ou plusieurs charges. Prenant l'exemple d'une attaque à l'aide d'un ver informatique (premier vecteur d'attaque), celui-ci utilise une attaque de type buffer overflow (deuxième vecteur d'attaque) afin d'installer un code malicieux généralement un cheval de Troie (première charge) au niveau de la machine cible. Ce cheval de Troie aura comme charge finale l'installation d'une porte dérobée (backdoor en anglais) au niveau de la machine (deuxième charge), qui donnera un accès à distance à l'attaquant.

2.4 Mécanismes de protection contre les attaques informatiques

Afin de remédier, aux attaques informatiques, ou au moins réduire leur risque, différents outils peuvent être utilisés tels que les systèmes cryptographiques, les pare-feux, les systèmes de détection d'intrusion, ainsi que les systèmes anti-malwares.

2.4.1 Les systèmes cryptographiques

La cryptologie est une science qui s'intéresse à la conception et l'évaluation des méthodes et techniques pour la protection de l'information (la protection de l'information englobe la confidentialité, l'intégrité, l'authenticité, etc.) [Pieprzyk et al., 2013]. Ainsi, la cryptographie est l'une des disciplines de la cryptologie qui s'intéresse à la conception d'algorithmes, protocoles, et de systèmes pour la protection de l'information contre une menace spécifique [Stamp, 2011, Pieprzyk et al., 2013]. En effet, l'action de crypter une information signifie la rendre compréhensible que pour la personne possédant la clé de cryptage/décryptage. Par rapport aux types d'attaques présentées dans la section précédente, le sniffing peut être évité en utilisant la cryptographie, à condition d'utiliser des clés de cryptage longues (256, 512 bits et plus). Par ailleurs, la cryptographie est aussi utilisée par les créateurs de malwares afin d'obscurcir (rendre illisible) certaines portions du code malicieux (voir chapitre 3).

2.4.2 Les pare-feux

Un pare-feu (en anglais Firewall) peut être un système logiciel, ou matériel, ou une combinaison des deux qui permet le filtrage des paquets dans les deux sens des connexions réseau

(entrantes et sortantes) [Kizza, 2015]. Le but d'un pare-feu est de fournir une protection contre les connexions non autorisées effectuées par les applications installées sur les machines locales avec des réseaux externes ou Internet, et protéger un réseau local contre les attaques provenant d'un réseau externe ou Internet [Gadelrab, 2008, Kizza, 2015]. La figure 2.2 illustre l'exemple où un pare-feu est installé entre un réseau privé et le réseau Internet.

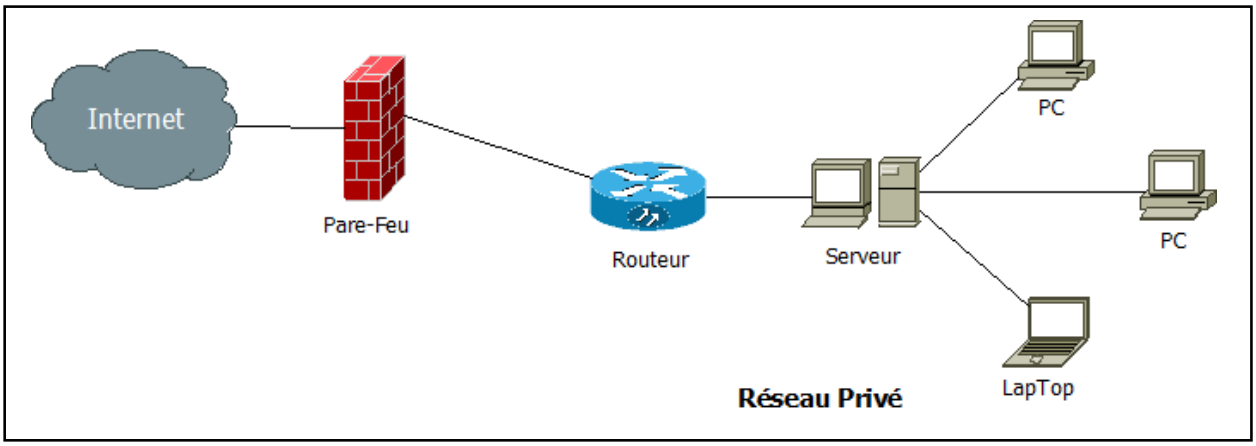

FIGURE 2.2 – Exemple d'un pare-feu installé entre un réseau privé et Internet.

Selon le type de service de sécurité offert pour chaque service dans le modèle TCP (Trasmission Cotrol Protocol)/IP (Internet Protocol), les pare-feux peuvent être divisés en quatre types qui sont[Kizza, 2015] :

- Pare-feu de la couche application : offre des services tels que le cryptage, et la passerelle niveau application.
- Pare-feu de la couche transport : offre principalement la fonctionnalité de filtrage de paquets TCP, UDP (User Datagram Protocol), ICMP (Internet Control Message Protocol).
- Pare-feu de la couche réseau : Offre la fonctionnalité de filtrage NAT (Network Address Translation) et IP.
- Pare-feu de la couche liaison de données : offre la fonctionnalité de filtrage des adresses MAC (Media Access Control).

2.4.3 Les systèmes de détection d'intrusions (IDS)

Les systèmes de détection d'intrusions (IDS pour Intrusion Detection System) sont des outils logiciels qui ont pour objectif de surveiller l'activité des hôtes ou du réseau afin de détecter et signaler toute activité pouvant être considérée comme intrusive, et ce par analyse d'événements générés par différents services ou utilisateurs [Briffaut, 2007, Hauser, 2013].

Une intrusion peut être définit comme étant une faute malveillante interne, mais d'origine externe, résultant d'une attaque qui a réussi à exploiter une vulnérabilité. Elle est susceptible de produire des erreurs pouvant provoquer une défaillance vis-à-vis de la sécurité, c'est-à-dire une violation de la politique de sécurité du système[Sang, 2012]. On distingue deux grandes familles d'IDS qui sont les IDS classiques et les IDS collaboratif (CIDS).

Les IDS Classiques

L'architecture classique d'un IDS est composée de trois éléments essentiels qui sont le capteur, l'analyseur, et le manager [Hiet, 2008]. Un ou plusieurs capteurs associés à un analyseur forment ce qu'on appelle une sonde de détection d'intrusion. Suivant la nature des informations récoltées par le capteur, les IDS peuvent être divisés en deux catégories qui sont les IDS systèmes (HIDS pour Host-based IDS) et les IDS réseau (NIDS pour Network-based IDS). Selon la méthode d'analyse employée par l'analyseur pour détecter la présence de menaces, on distingue les IDS basés sur l'analyse d'anomalies (Anomaly detection) et les IDS basés sur les scénarios d'attaques (Misuse detection) [Zaki and Sobh, 2004, Gadelrab, 2008, Wu and Banzhaf, 2010]. La figure suivante (figure 2.3) illustre un exemple d'architecture d'un IDS classique.

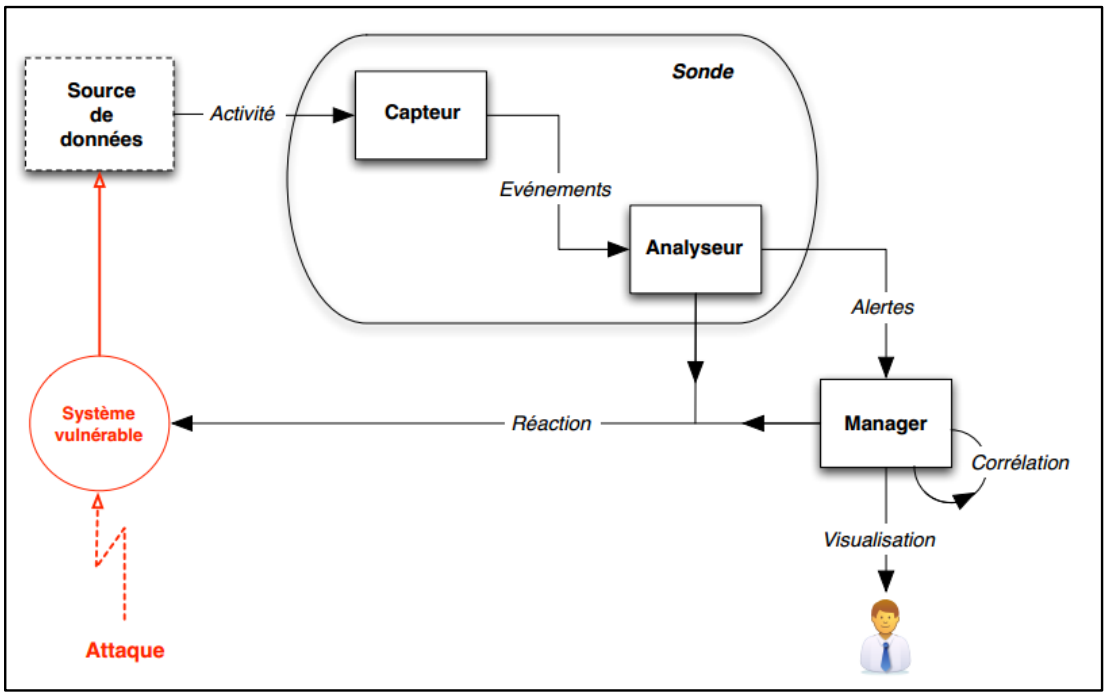

FIGURE 2.3 – Architecture d'un IDS classique [Hiet, 2008].

– Le capteur : A pour but la collecte des informations sur l'activité du système à partir de

différentes source de données tels que les interfaces réseaux, les journaux systèmes, etc. Il transmet ensuite ces informations (brutes ou pré-traitées) à l'analyseur sous forme de séquences d'événements qui l'informe de l'évolution de l'état du système [Hiet, 2008]. On distingue deux types de capteurs selon les types d'IDS présentés précédemment :

  - Les capteurs systèmes sont employés par les IDS systèmes (HIDS) et ont pour but de collecter des données sur l'activé de l'hôte, tel que, des changements apportés aux fichiers systèmes, les connexions réseaux effectuées, etc. Ceci est effectué notamment par l'analyse des journaux d'audit système, ou par celui des appels systèmes invoqués par les applications.

  - Les capteurs réseaux sont utilisés par les IDS réseaux (NIDS), et ils collectent les données en analysant le trafic réseau entre les machines.

- L'analyseur : il a pour but de filtrer et de qualifier les événements fournis par le capteur. Si l'analyseur juge qu'un ensemble d'événements contient des caractéristiques d'une activité malveillante, il lève une alerte qui sera transmise au manager. Il existe deux grandes méthodes d'analyses qui sont l'analyse d'anomalies (Anomaly detection) et l'analyse par scénarios d'attaques (Misuse detection).

  - L'analyse d'anomalies, aussi appelée analyse comportementale, consiste à établir au préalable un modèle représentant un comportement normal, et considère comme malicieux tout autre comportement ne relevant pas du comportement normal. En effet, une tentative d'intrusion, implique la modification du comportement d'un service, d'une application ou même d'un utilisateur, telle qu'une utilisation anormale de ressources, l'accès à des fichiers ou une utilisation anormale du réseau [Labbe, 2005, Briffaut, 2007].

  - L'analyse par scénarios d'attaques, ou bien analyse par signatures, consiste à générer une signature (scénario) pour chaque attaque connue. Ces signatures sont stockées dans une base de signatures d'attaques. Le processus de détection d'une attaque consiste à utiliser un algorithme de recherche de motifs, comparant le flux d'événements aux signatures contenues dans la base [Briffaut, 2007].

- Le manager : Lors de la réception d'une alerte en provenance de l'analyseur, le manager se charge de la transmission de celle-ci à l'opérateur (l'administrateur réseau).

Il peut également réaliser les fonctions de corrélation d'alertes, dans la mesure de leur disponibilité. Enfin, il peut assurer le traitement de l'incident, par exemple au travers du confinement et de l'éradication de l'attaque, la restauration du système, ainsi que le diagnostic et la contre-mesure [Hiet, 2008].

Les IDS Collaboratifs

Contrairement aux IDS classiques qui agissent de manière isolée, les IDS collaboratifs (CIDS) coopèrent entre eux afin d'aboutir à une décision collaborative sur la présence ou non d'une tentative d'intrusion. Les CIDS ont été introduits dans le but de remédier aux problèmes liés aux faux positifs dont souffrent les IDS classiques [Wu et al., 2003]. En outre, et contrairement aux IDS classiques, les CIDS ont la capacité de détecter les attaques coordonnées, vu leur potentiel pour détecter les intrusions qui se produisent dans l'ensemble du réseau Internet en même temps en mettant en corrélation des signatures d'attaques entre différents sous-réseaux [Zhou et al., 2010]. Étant donné que nous proposons dans cette thèse une approche collaborative pour la détection des malwares, et étant donné les différents points en communs avec les CIDS, nous différons les détails de ces derniers au chapitre 5.

2.4.4 Les anti-malwares

Un système de détection de malwares (antimalware) est un programme qui permet de détecter la présence de codes malicieux sur un ordinateur et de les supprimer. L'éradication d'un malware se fait de plusieurs façons qui sont [Kizza, 2015] :

- La désinfection du fichier infecté en supprimant le code malveillant.
- La suppression du fichier infecté entièrement.
- La mise en quarantaine du fichier infecté, ce qui implique son déplacement vers un endroit où il ne pourra pas être exécuté.

Les antimalwares utilisent différentes techniques pour la détection des codes malicieux, telles que les techniques basées signatures (qui sont souvent utilisées par les antivirus commerciaux), les techniques comportementales, et les techniques heuristiques [Siddiqui et al., 2008a, Bazrafshan et al., 2013]. Dans le chapitre suivant nous allons revenir en détails sur ces trois différentes techniques de détection de code malveillants.

2.5 Conclusion

Dans ce chapitre, nous avons survolé le domaine de la sécurité informatique en abordant brièvement les différents concepts relatifs à ce domaine. Nous avons essayé de mettre l'accent sur les différents types d'attaques existants, ainsi que les différentes techniques pour la protection des systèmes informatiques contre ces attaques. Il est à noter dans cette conclusion, la grande diversité des attaques qui peuvent être menées contre les systèmes informatiques, et par conséquence, l'apparition d'un nombre élevé d'approches, de méthodes, et de systèmes pour leur détection et éradication. Néanmoins, il faut noter également que la vulnérabilité des codes, et la vulnérabilité humaine restent les plus fréquentes, et auxquelles on doit accorder plus d'importance si on souhaite voir assurer un niveau élevé de sécurité. Dans le chapitre suivant, nous allons nous intéresser au domaine de la virologie informatique, en présentant les différents types de malwares, leurs différents modes d'infection, ainsi que les mécanismes existants pour la protection contre ce type de menaces.

Chapitre 3

Virologie Informatique

Sommaire

3.1 Introduction

Si aujourd'hui la sécurité informatique consiste en une préoccupation majeure pour les administrateurs des systèmes, et si les différentes attaques sont prises en compte lors de la mise en place de politiques et d'outils pour la sécurisation des ordinateurs, les attaques via les codes exécutables, restent les plus redoutables et dont la communauté de sécurité prête

plus d'attention, comparée aux autres menaces. Initialement le mot "Virus" est utilisé pour rapprocher le concept de malware aux aspects biologiques et médicaux qui peuvent altérer néfastement la santé de l'homme. Cependant, et avec la diversification des formes et des fonctionnalités, de ces codes malveillants, on leur a attribué le qualificatif "malware", qui signifie tout software dont le développeur a des intentions malveillantes. Dans la sécurité informatique, il s'agit de la menace la plus redoutée, du fait qu'il est toujours difficile de distinguer un code malware d'un code sain. D'autre part, quand un malware est exécuté dans la zone locale d'un système informatique, ceci consiste un préjudice extrême, car le contrôle du système en entier passe entre les mains du Hacker.

Le terme 'Malware' est un terme générique qui se réfère à tout programme s'exécutant sur une machine en y accomplissant des actions à l'insu de l'utilisateur. Selon la définition proposée par Salehi et al. [Salehi et al., 2014], *'un malware est un programme informatique qui a été élaboré afin de pénétrer ou endommager un système informatique sans la permission de l'utilisateur'*. Durant ces vingt dernières années, le nombre des malwares n'a pas cessé d'augmenter. L'un des facteurs majeurs ayant contribué à cette prolifération est le développement des réseaux de communication, ainsi que la banalisation de l'accès à Internet. La grande famille des malwares regroupe les virus, les vers, les chevaux de Troie, etc (voir section 3.3). De nos jours, les frontières entre ces types de malwares sont presque inexistantes. En effet, un malware peut regrouper plusieurs fonctionnalités caractérisant différents types de malwares, c'est ce qu'on appelle une menace combinée (Blended Threat en anglais) [Erbschloe, 2004, Aycock, 2006, Malin et al., 2011].

Dans ce chapitre, nous allons présenter les différents concepts liés au domaine de l'analyse et de la détection des malwares, en abordant certains aspects fondamentaux, tels que :

- Les origines des malwares.
- Les différents types de malwares et leurs différents modes d'attaques (d'infection).
- Les méthodes d'obscurcissement utilisées par les malwares.
- Les différentes techniques pour l'analyse et la détection des malwares.

3.2 Historique des malwares

Comme mentionné dans l'introduction générale de ce manuscrit, les origines des codes malicieux remontent aux années 1940 avec les travaux de John Von Neumann sur les automates auto-réplicatifs. Dans leurs débuts, ce que nous appelons aujourd'hui malwares, n'avaient pas un aspect malicieux. En effet, ils étaient créés à des fins expérimentales, pour divertissement, ou même par erreur. Par ailleurs, le peu d'entre eux ayant été créer à des fins malicieuses se

contentaient de causer une nuisance par l'affichage d'un message sur l'écran de la machine infectée ou la saturation de celle-ci par une réplication excessive du programme comme ce fut le cas avec *Creeper* et *Wabbit* (créés en 1971 et 1974 respectivement) [Rajesh et al., 2015]. En 1981, un programme nommé *Elk Cloner* [Aycock, 2006], s'est propagé dans les machines dotées d'un système d'exploitation Apple DOS [1] en affichant un poème sur l'écran de la machine infectée. *Elk Cloner* était le premier programme à se propager via un support de stockage (disquette). En 1983 Frederick Cohen utilise le terme Virus pour définir les programmes informatiques autoreproducteurs. En 1986, deux informaticiens d'une petite entreprise d'informatique pakistanaise *'Brain Computer Services'*, créèrent involontairement le premier virus MS-DOS connu sous le nom de '*The Pakistani*' (le pakistanais). Leur but était de protéger leurs logiciels du piratage. *The Pakistani* se propageait via disquette et replacer son étiquète par les informations personnelles des deux frères (nom et prénom, adresse, numéro de téléphone, etc.) [Highland, 1997, Zou et al., 2006, Kizza, 2015]. En 1987 Fred Cohen établit les bases pour les techniques de défense contre ces programmes, en démontrant en même temps qu'il n'existe aucun algorithme assez efficace pour détecter tous les virus [Cohen, 1987].

Durant toute cette période, les codes malicieux se résumaient à des virus qui se répliquent dans une même machine, ou de machine en machine via des disquettes. En 1986, une nouvelle génération de codes malicieux a vu le jour, à savoir les chevaux de Troie (Trojan Horse en anglais). *PC-Write Trojan* à été conçu en le maquillant de façon à faire croire aux utilisateurs qu'il s'agissait d'une nouvelle version du logiciel de traitement de texte PC-Write. Ce dernier était considéré comme étant le premier Shareware [2], ce qui a facilité la propagation de codes malicieux. Une fois exécuté, ce malware efface toutes les données contenues dans le disque dur de la machine [Wang, 2006].

Ce n'est qu'à la fin des années quatre vingt que les malwares ont commencé à causer des dégâts notables avec la découverte de *Morris* en 1988. *Morris* a été créé par erreur par Robert Tappan Morris qui à la base voulait comptabiliser le nombre d'ordinateurs connectés à Internet. À cause d'un bug dans son code, le programme s'est propagé via le réseau Internet à une vitesse surprenante infectant des milliers de machines en quelques heures. *Morris* exploitait une vulnérabilité de type Buffer Overflow dans les programmes SendMail et Fingered sous Unix. Les dommages causés par ce qu'on a appelé *'le crash d'Internet de 1988'* ont été estimés à des dizaines de millions de dollars [Tittel, 2004]. La découverte de *Morris* a également marqué l'avènement d'une nouvelle catégorie de programmes autoreproducteurs, à savoir les vers informatiques.

1. Apple DOS est un système d'exploitation développé pour les machines Apple II.
2. Le terme Shaware est utilisé pour désigner tout logiciel pouvant être partagé gratuitement.

Au milieu des années quatre-vingt-dix, et avec la popularité croissante des systèmes d'exploitation Microsoft Windows, on a assisté à la naissance du premier macrovirus à savoir *Concept*. Ce dernier, découvert 1995 infecte les documents du logiciel de traitement de texte Microsoft Word, et se propage de machine en machine par le partage des fichiers infectés.

En 1998, le virus *Tchernobyl* aussi appelé CIH a détruit le programme BIOS des cartes mères de milliers d'ordinateurs, les rendant inutilisables, et causant ainsi, à titre d'exemple pour la Corée du Sud à elle seule des dégâts estimés a près de 250 millions d'euros [Filiol, 2009]. Un an plus tard, un nouveau type de vers a vu le jour. Ces derniers se propagent via courriel et utilisent des techniques d'ingénierie sociale afin d'infecter les machines. *Melissa*, qui a été découvert en 1999, se propage via MS Outlook en envoyant une copie de lui-même par courriel avec comme attachement un fichier Word infecté. Les dégâts du virus *Melissa* ont été estimé à plus de 80 millions de dollars [Kizza, 2015]. *Mellissa* a été suivi un an plus tard par *LoveLetter*, ce dernier, se propage de la même façon sauf que le courriel contient un attachement nommé *LoveLetter* qui est en fait un fichier de script Visual Basic. Une fois exécuté, il déclenche la charge du virus ce qui engendre sa réplication dans des fichiers systèmes critiques ainsi que sa propagation [Kizza, 2015].

CodeRed a été découvert en 2001, et c'était le premier malware qui avait l'aptitude à se propager via le réseau Internet sans aucune intervention de l'utilisateur. Ce malware exploite une vulnérabilité de type buffer overflow dans un utilitaire d'indexation installé par défaut dans les serveurs web Microsoft IIS. *CodeRed* a causé la mise hors service de milliers de serveurs dans le monde [Zou et al., 2002]. Il a été suivi deux ans plus tard par SQL Slammer qui a infecté plus de 70000 machines en moins de dix minutes.

Une nouvelle ère pour les codes malicieux a commencé au milieu des années 2000. Cela c'est traduit par la découverte en 2004 de *CabirWorm* [Filiol, 2009], le premier malware infectant les téléphones mobiles ainsi que le ver *Koobface* en 2005 [Timm and Perez, 2010], qui était le premier malware se propageant via les réseaux sociaux.

Durant l'année 2007, la découverte de *Zeus* (aussi appelé Zbot) a permis de constater que les motivations des créateurs de malwares ont évolué. Ces derniers sont devenus des cybercriminels motivés par le gain financier. *Zeus* qui est un cheval de Troie, a pour but de collecter toutes informations pouvant être revendues ou utilisées, et plus spécialement les informations des comptes bancaires. Ces dernières, étaient collectées en interceptant les informations échangées via le navigateur web avec des sites de commerce électronique ou des banques, ou bien à l'aide d'un Keylogger.

En 2008, on a découvert *Conficker*, ce malware exploite une vulnérabilité de type buffer overflow dans les services du serveur sous MS Windows. Le ver peut causer l'arrêt des services de sécurité du système d'exploitation, et bloquer l'accès aux sites web des antivirus. En outre,

il permet de recruter la machine infectée dans un réseau de Botnets qui sera utilisé pour mener des attaques coordonnées [Kramer and Bradfield, 2010].

L'année 2010 a marquée l'avènement de ce qu'on a appelé 'cyber guerre', avec la découverte de *Stuxnet* [Erbschloe, 2004, Alam et al., 2014]. *Stuxnet* a été destiné à mettre hors services les systèmes industriels SCADA qui sont largement utilisés dans des infrastructures critiques y compris dans les réacteurs nucléaires. Après sa découverte, des rumeurs se sont propagées faisant allusion que ce dernier a été spécialement conçu pour déstabiliser le programme nucléaire Iranien. *Stunext* est vu comme une cyber-arme qui peut causer des dégâts colossaux pour des populations entières[Malin et al., 2011].

Après la découverte de *Stuxnet*, on a réalisé que les attaques par malwares pouvaient avoir des répercussions catastrophiques. Les malwares sont devenus des armes à part entière qui sont utilisés par des organisations de crime organisées motivées par le profit. Des malwares de type Trjan Banker, tel que *SpyeEye* découvert en 2011, a permit le vol d'informations de comptes bancaires à partir de machines compromises ou bien des Smartphones. Les malwares BlackPOS, FrameworkPOS, et Backoff, qui ont été découverts entre 2013 et 2014, sont des malwares conçus pour s'attaquer aux machines des points de vente (POS, Point of Sale en anglais). Ces derniers ont été à l'origine en 2014 du vol de plusieurs dizaines de millions de données relatives aux informations des cartes de crédit [Marschalek et al., 2014]. *Locky*, qui est un malware de type Rançongicel (voir section 3.3.6) a été découvert au mois de février 2016 [3]. Ce dernier crypte les fichiers sauvegardés sur la machine de la victime et change leur extension à '.locky', afin de restituer le contenu des fichiers, la victime doit se procurer pour une centaine de dollars la clé de déchiffrement via le web obscure (darknet) [4].

Durant les dix dernières années, les malwares sont devenus de plus en plus sophistiqués, ils utilisent des méthodes d'infection très complexes exploitant plusieurs vulnérabilités en même temps. Ils sont aussi devenus très difficile, à détecter, en employant des méthodes avancées pour l'obscurcissement de code. En outre, les motivations derrière leur création ont aussi évoluées, elles peuvent être purement financières ou même politiques.

3.3 Les types de malwares

Dans cette section, nous allons présenter les différents types de malwares, ainsi que leurs modes opératoires. Les malwares peuvent être divisés en deux grandes catégories qui sont les codes simples et les codes autoreproducteurs [Filiol, 2009]. Contrairement aux codes simples, les codes autoreproducteurs ont la capacité à se dupliquer et à se propager. Dans cette section,

3. Semantec Labs : `http://www.symantec.com/connect/blogs/locky-ransomware-aggressive-hunt-victims`
4. TechTarget, darknet (2011) : `http://searchnetworking.techtarget.com/definition/darknet`

nous allons présenter deux types de malwares autoreproducteurs qui sont les vers et les virus ainsi que certains malwares simples comme les chevaux de Troie, les portes dérobées, les Rançongiciels, ainsi que les Keyloggers.

Il s'agit ici d'une typologie communément rencontrée dans la littérature. Cependant, rien n'empêche que ces types se chevauchent. En effet, un Keylogger ou un Rançongicel peuvent être griffés sur des vers, et deviennent par conséquence autoreproducteurs. Cependant, c'est difficile d'imaginer un cheval de Troie autoreproducteur, du fait qu'il s'agit d'une application entière dotée d'une interface, et nécessairement visible pour l'utilisateur, et de ce fait difficile de considérer sa duplication et sa propagation.

3.3.1 Les Vers

Les vers informatiques (Worms en anglais) sont des programmes malveillants, qui une fois installés dans une machine, utilisent un réseau pour envoyer des répliques d'eux même à d'autres machines [Erbschloe, 2004]. Les vers peuvent perturber le trafic dans un réseau, supprimer des fichiers, envoyez des documents par courriel, ou bien installer d'autres codes malicieux tels que des Backdoors (voir section 3.3.4) sur les machines infectées. Une fois une nouvelle machine infectée, celle-ci commence à son tour à rechercher d'autres machines à infecter. Et ainsi, le ver va continuer de se répliquer jusqu'à ce qu'un mécanisme arrête le processus [Erbschloe, 2004].

Le mode opératoire d'un ver passe par différentes phases qui impliquent l'utilisation de différents mécanismes. Weaver et al. [Weaver et al., 2003] ont proposé une taxonomie pour la classification des vers selon plusieurs critères, qui sont : la méthode de découverte de victimes, le mode de transmission, le mode d'activation, la nature de la charge, et la motivation des attaquants. J. Nazario [Nazario, 2004] a proposé une décomposition des vers en cinq modules principaux, qui sont : la méthode de reconnaissance des cibles, le mode d'attaque, les moyens de communication, la commande, et l'intelligence. P. Szor [Szor, 2005] propose une structure dite générique d'un ver qui inclut les composants suivants : le localisateur de cibles, le module transfert, le module de communication, le gestionnaire du cycle de vie, la charge, et le module de suivi.

En se basant sur les trois décompositions présentées ci-dessus, nous allons proposer une structure générique des vers qui comporte les composants jugés indispensables, à savoir : le module de recherche de cibles, le module de propagation, le module d'activation, et la charge.

- **Le module de recherche de cibles :** Ce module permet au ver de découvrir des victimes potentielles, ce qui s'avère être d'une importance primordiale afin d'assurer sa propre propagation. Selon l'espace dans lequel les vers recherchent leurs victimes, on distingue plusieurs types de vers qui sont, les vers Internet (trouvent leurs victimes

dans l'espace IP), les vers de courriel (trouvent leurs victimes dans l'espace des adresses e-mail), les vers P2P (trouvent leurs victimes dans l'espace des voisins des réseaux pair-à-paire), et les vers de messagerie (trouvent leurs victimes dans l'espace des contacts de messageries instantanée)[Brand, 2010].

La méthode de recherche des cibles se fait généralement de façon active ou passive.

- Recherche active : Ce mode consiste à rechercher des victimes dans l'espace de propagation. Dans le cas d'un ver se propageant via e-mail, ce dernier va tenter d'obtenir le carnet d'adresses de l'utilisateur et envoyer des copies de lui-même à tous les contacts. Les vers se propageant via un réseau utilisent une méthode scan. Le scan s'effectue de deux façons, à savoir, aléatoirement ou séquentiellement. Les vers utilisant une méthode de scan aléatoire sélectionnent une adresse IP parmi les adresses de l'espace IP et essaient de se connecter à cette machine. Dans le scan séquentiel, le ver dispose d'une liste d'adresses IP de toutes les machines qui peuvent être vulnérables dans un réseau donné. Celles-ci seront parcourues une-par-une par le ver en faisant des tentatives de connexion à chaque fois.
- Recherche passive : Les vers utilisant cette technique ne recherchent pas activement leurs victimes mais plutôt ils sont en attente que des machines viennent se connecter à la machine infectée pour qu'ensuite le ver infecte cette machine à son tour.

- **Module de propagation :** Une fois la cible localisée, le ver va commencer sa phase de propagation, à savoir le transfert de son code vers la machine cible. Pour cela, il existe trois différentes approches qui sont l'autoportée, le transfert en deux phases, et l'intégration [Weaver et al., 2003]. L'autoportée consiste à envoyer la totalité du code du malware lors de la communication initiale avec la machine cible. Le transfert en deux phases s'effectue lorsqu'un ver télécharge une petite partie de son code lors de la communication initiale qui s'occupera ensuite du téléchargement du reste du code. L'intégration consiste à transmettre le code du malware en l'intégrant dans le canal de communication, soit en modifiant la structure du message, soit en les remplaçant complètement.

- **Module d'activation :** Il est censé conditionner le lancement de l'action que le malware est supposé accomplir (sa charge finale). Il dépend de la façon avec laquelle le ver sera exécuté dans la machine où il est téléchargé. On distingue deux méthodes possibles pour l'activation d'un malware, à savoir l'activation manuelle ou automatique. Par activation manuelle on sous-entend la nécessité de l'intervention de l'utilisateur soit directement en exécutant le ver, soit indirectement en effectuant une action pouvant

activer le ver tel le redémarrage de la machine, l'ouverture de session, etc. L'activation automatique se fait par l'exploitation de vulnérabilités dans un ou plusieurs services de la machine infectée.

– **La charge (Payload) :** Elle contient l'ensemble des séquences d'actions qu'un malware est censé accomplir. Il peut s'agir d'actions pour la destruction de fichiers, l'installation d'un code malicieux, pilotage d'applications, etc.

3.3.2 Les Virus

Tel un virus biologique, un virus informatique utilise un programme hôte afin de se reproduire et en infecter d'autres par la modification de leur contenu. Contrairement aux vers, les virus ne sont pas autonomes et ont besoin du fichier hôte pour se propager dans un même système, ou d'infecter un support amovible pour se propager d'une machine à une autre [Aycock, 2006, LeDoux and Lakhotia, 2015]. Selon le type de fichiers ciblés, les virus informatiques peuvent être répartis en trois catégories qui sont : les virus du secteur de boot, les virus de documents, et les virus d'exécutables [Kephart et al., 1997].

– **Virus du secteur de boot :** le secteur de boot (secteur d'amorçage), est le premier secteur dans un disque. Il a pour rôle de sauvegarder un bout de code, et des données nécessaires au démarrage du système d'exploitation. Un tel virus remplace le secteur d'amorçage par une copie de lui-même et déplace le secteur original vers un autre emplacement du disque. Cela va causer le chargement du virus à chaque démarrage du système.
– **Virus de documents :** ce type de virus infecte différents types de documents tels que les fichiers scriptes (ex. VBS, JavaScript), les documents Microsoft Office, les fichiers PDF, etc. Un virus de documents est activé par l'interpréteur du contenu de façon native dans l'application associée au format de fichier [Filiol, 2009].
– **Virus d'exécutables :** ce type de virus s'attaque aux fichiers d'application ainsi qu'aux fichiers exécutables systèmes tels que les fichiers .exe, .com, .dll, .sys, .drv, .bin, etc.

L'infection des fichiers par un virus se fait principalement à l'aide de cinq techniques qui sont : le recouvrement, l'écrasement, l'entrelacement, l'accompagnement de code, et par modification du code source [Okamoto and Ishida, 2000, Filiol, 2009].

– **Infection par recouvrement de code :** C'est une infection très répondue parmi les

virus infectant les programmes MS-DOS [Okamoto and Ishida, 2000]. Elle consiste à ajouter le code du virus à la fin du fichier hôte (cible), et à en modifier son entête de façon à ce que le code malicieux s'exécute en premier, ensuite le code d'origine est à son tour exécuté (figure 3.1), ce qui permet de dissimuler le virus [Filiol, 2009].

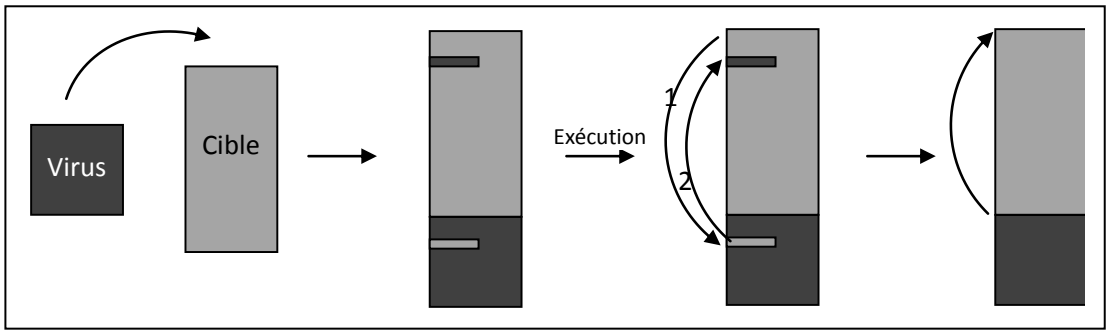

FIGURE 3.1 – Méthode d'infection par recouvrement de code [Filiol, 2009].

– **Infection par écrasement de code :** Cette technique consiste à écraser le contenu du fichier hôte et ainsi le fichier d'origine n'est plus disponible (figure 3.2). De ce fait, il est impossible de réparer le fichier infecté, mais il sera facile de l'identifier par un logiciel anti-virus [Aycock, 2006].

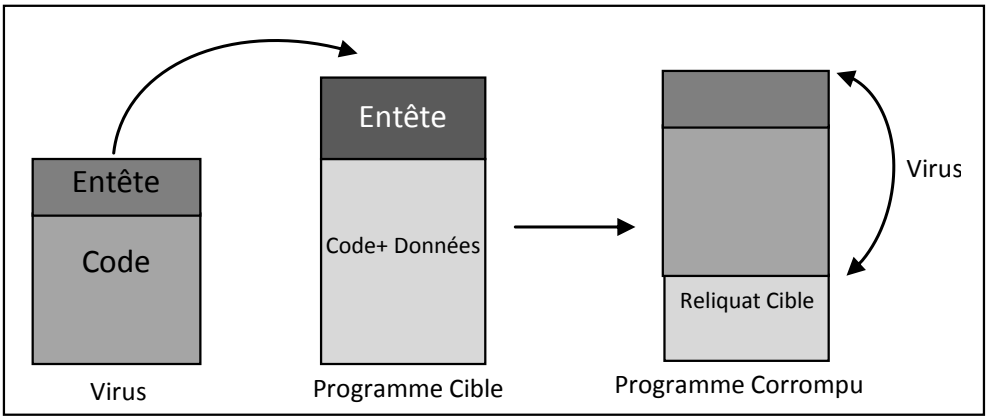

FIGURE 3.2 – Méthode d'infection par écrasement de code [Filiol, 2009].

– **Infection par entrelacement de code :** c'est une technique d'infection très sophistiquée. Elle a été utilisée entre autre par le virus Tchernobyl (CIH) cité précédemment [Filiol, 2009]. Ce type d'infection est spécifique aux virus infectant les codes exécutables PE (Portable Exécutable, voir section 4.2.2), et cela en insérant des fragments du code malicieux dans les espaces inoccupés du fichier, comme l'illustre la figure (figure 3.3).
– **Infection par accompagnement de code :** Le virus duplique son code, non pas en l'insérant dans le code cible, mais en créant un fichier supplémentaire qui va «accompa-

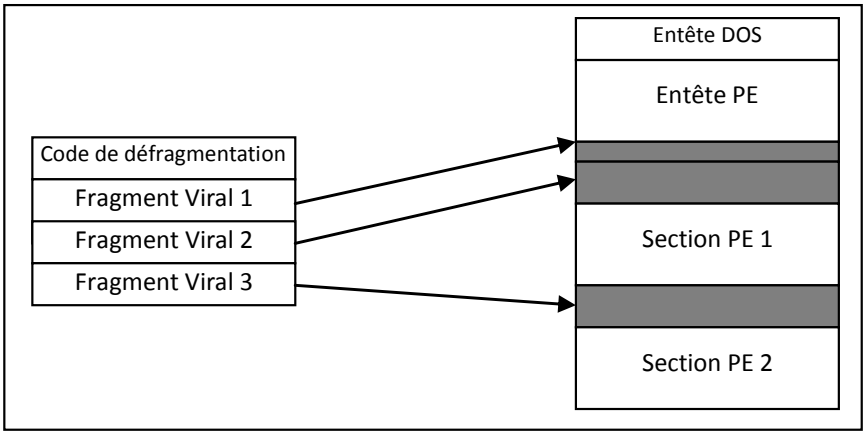

FIGURE 3.3 – Méthode d'infection par entrelacement de code [Filiol, 2009]

gner» la cible (figure 3.4). Lors de l'exécution du programme cible infecté, c'est la copie virale contenue dans ce fichier supplémentaire qui est en réalité exécutée en premier, permettant au virus de se propager, ensuite, ce dernier exécute lui-même le programme cible légitime qu'il accompagne [Filiol, 2009].

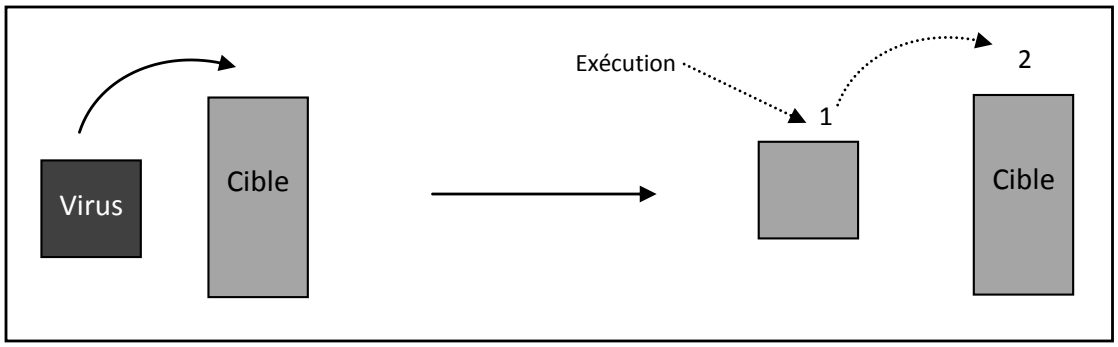

FIGURE 3.4 – Méthode d'infection par accompagnement de code [Filiol, 2009]

– **Infection par modification du code source :** Dans ce type d'infection, le virus tente d'infecter les fichiers contenant le code source du programme (*.C, *.Pas, *.ASM, etc.), et non pas le fichier exécutable. Une fois le code viral inséré et le code source compilé, le code exécutable en résultant -une fois exécuté- se mettra à infecter d'autres fichiers source du même type [Ludwig, 1998, Filiol, 2009].

3.3.3 Les chevaux de Troie

Un cheval de Troie (en anglais Trojan Horse, abrévié Trojan) est un programme qui peut paraitre inoffensif (bénin), généralement accompli les tâches pour lesquelles il a été crée (ex.

Un lecteur multimédia). Cependant, il exécute également des tâches à l'insu de l'utilisateur [Aycock, 2006]. Les chevaux de Troie peuvent offrir au hacker la possibilité d'effectuer différentes tâches à distance dans la machine infectée, tels que la suppression des fichiers, le téléchargement de fichiers, modifier les paramètres du registre, arrêter l'exécution de certaines applications telles que les anti-virus, etc [Erbschloe, 2004]. Le mode opératoire d'un Trojan inclut l'utilisation de deux modules qui sont le module serveur (le programme lui-même) et le module client qui est utilisé pour prendre le contrôle des machines infectées par le module serveur [Filiol, 2009]. La figure suivante (figure 3.5) illustre le fonctionnement d'un cheval de Troie.

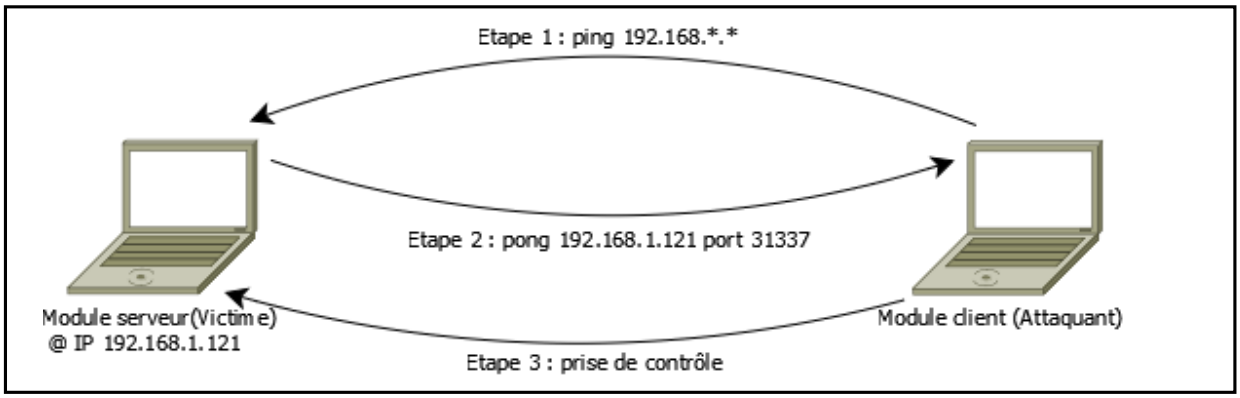

FIGURE 3.5 – Mode opératoire d'un cheval de Troie [Filiol, 2009]

Le mode d'infection d'un Trojan commence par l'installation du programme malicieux (module serveur) par l'utilisateur après l'avoir téléchargé ou qu'il ait reçu par courriel pensant qu'il s'agit d'un programme bénin. Une fois le module serveur installé, le module client (attaquant) attend la réception d'un message (contenant l'adresse IP et le port TCP ou UDP) de la part de la machine infectée, ou bien il procède à un scan (par des requêtes Ping) des machines infectées. Une fois le message reçu, le module client prend le contrôle de la machine infectée qui se traduit par l'exécution de commandes à distance sur la machine infectée. Selon leur charge, on peut distinguer un grand nombre de variantes des chevaux de Troie, tels que les Trojan-Dropper, Trojan-Downloader, Trojan-Spy, etc. [Brand, 2010].

3.3.4 Les portes dérobées

Une porte dérobée (Backdoor en anglais) peut être définie comme étant *un programme qui permet à des attaquants de contourner les mesures de sécurité d'un système, afin d'y obtenir l'accès* [Skoudis and Zeltser, 2004]. L'installation d'une porte dérobée requiert un accès à la machine cible. De ce fait, l'installation doit se faire soit manuellement en transmettant le

malware à l'utilisateur (ex. par courriel) et le convaincre d'installer le programme (méthode basée sur l'ingénierie sociale), soit automatiquement en utilisant un autre code malicieux tel un ver, ou un cheval de Troie. Une fois installée, la porte dérobée va donner un accès à l'attaquant. Cela se fait généralement par l'ouverture d'un port réseau et en acceptant des commandes à provenance de la machine de l'attaquant généralement via le protocole HTTP distant via Internet [Szor, 2005, Sikorski and Honig, 2012]. L'attaquant peut ensuite effectuer différentes tâches sur la machine infectée telles que l'obtention de privilèges administrateur, et le contrôle à distance via l'exécution de commandes [Skoudis and Zeltser, 2004].

3.3.5 Les Keyloggers

Le rôle d'un Keylogger est de capturer les caractères saisis sur un clavier d'une machine. Les keyloggers sont généralement programmés pour s'exécuter au démarrage du système, et enregistrent dans des fichiers (journaux) tout caractère saisi que ça soit dans les courriels, les messages, les documents et même les noms d'utilisateurs et mots-passes. Cela se fait généralement à l'aide d'une fonction Hook [Erbschloe, 2004, Aycock, 2006]. Un keylogger peut être installé par un administrateur dans une entreprise afin de contrôler les activités des employés, comme il peut être installé par un Hacker afin d'obtenir des informations sur ses victimes. La figure 3.6 illustre le mécanisme de fonctionnement d'un Keylogger.

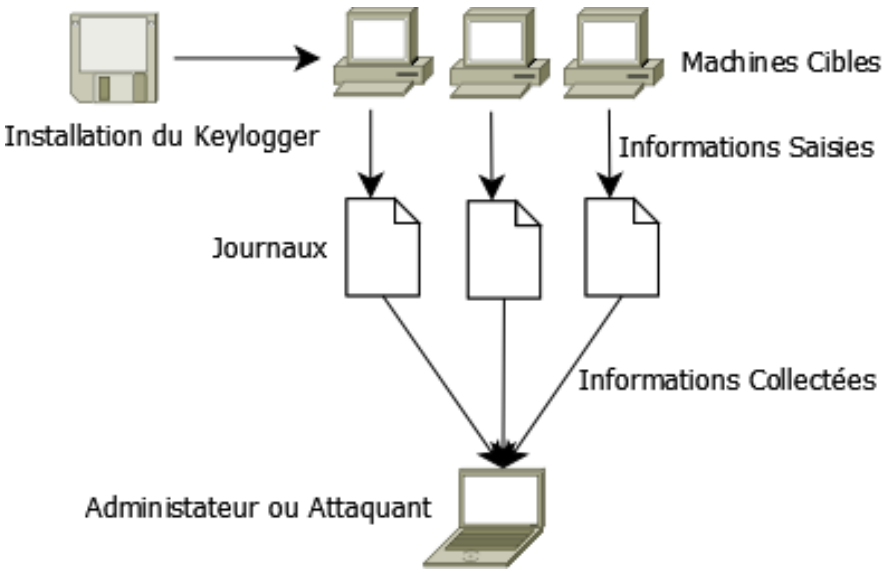

FIGURE 3.6 – Mode opératoire d'un Keylogger.

3.3.6 Les Rançongiciels (Ransomware)

Le terme Ransomwre est l'acronyme pour Ranson et Malware, et est utilisé pour définir tout type de malware conçu dans le but d'exiger une certaine somme d'argent en contrepartie de la restitution d'une information ou fonctionnalité dérobée [Gazet, 2010]. Par exemple, un Hacker peut utiliser un Ransomware qui va crypter des informations stockées dans la machine de la victime, et ensuite lui demander une certaine somme d'argent afin de les décrypter.

3.4 Les techniques d'obscurcissement

Afin de rendre leurs codes malveillants difficiles à détecter, les hackers utilisent des méthodes d'obscurcissement, qui se divisent en trois catégories et qui sont : la compression (en anglais Packing), le polymorphisme et le métamorphisme [O'Kane et al., 2011, Alam et al., 2014]. Dans ce qui suit, nous allons présenter ces différentes techniques.

3.4.1 La compression (Packing)

Le Packing signifie la compression d'un fichier exécutable, qui une fois lancé, va procéder à sa décompression avant son exécution effective. Cette technique a été développée dans un but légitime qui est la protection des logiciels commerciaux des risques de piratage, cependant elle a été utilisée par les créateurs de malwares afin de rendre leurs programmes difficiles à détecter, ainsi que pour faciliter leur propagation dans les réseaux en réduisant leurs taille [Han and Lee, 2009, O'Kane et al., 2011]. La figure 3.7 présente la structure d'un fichier PE avant et après sa compression.

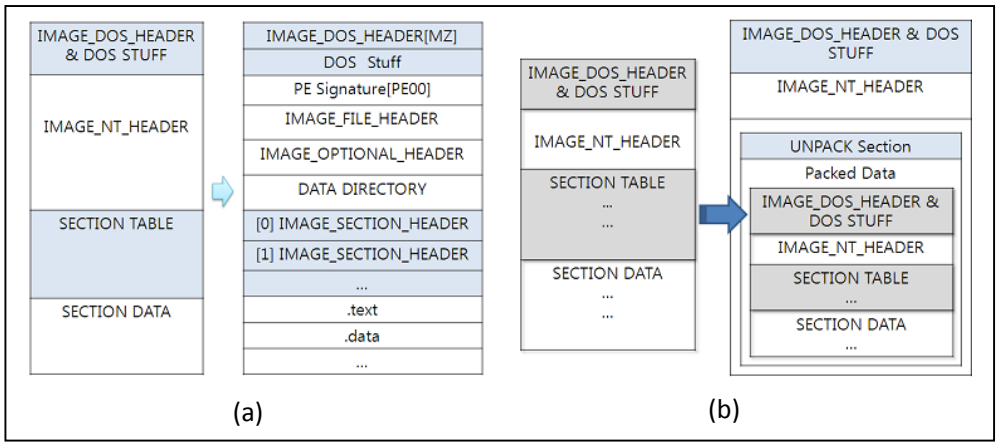

FIGURE 3.7 – Comparaison entre la structure de deux programmes PE, l'un non-compressé (a) et l'autre compressé (b) [Han and Lee, 2009].

La compression permet donc de changer la structure du code exécutable tout en conservant la même fonctionnalité, ce qui le rend difficile à détecter par les antivirus classiques.

3.4.2 Le polymorphisme

Les malwares dits polymorphes consistent principalement à crypter le programme cible en utilisant différentes clés cryptographiques et fournir en même temps le mécanisme de déchiffrement, dans le but de rendre certaines portions de leur code illisible [Li et al., 2011].

P. Okane et al. [O'Kane et al., 2011] présentent un exemple simplifié du fonctionnement d'un malware polymorphe. Selon cet exemple, le malware, après avoir été exécuté commence par décrypter le code du malware à l'aide de la clé de chiffrement qui est stockée au niveau du fichier malicieux. Le code est ensuite chargé en mémoire, ce qui engendre l'exécution du code malicieux (la charge). Une fois cette phase terminée, une nouvelle clé de chiffrement est générée. Elle est utilisée afin de crypter une nouvelle fois le code du malware. Le fait de générer une nouvelle clé, permet d'obtenir une nouvelle version du malware à chaque exécution.

3.4.3 Le métamorphisme

Un malware métamorphique, est capable de changer sa structure interne, en modifiant son code machine lors de son chargement en mémoire, avant de le réécrire à nouveau dans le fichier hôte, tout en conservant le même comportement (fonctionnalité) [O'Kane et al., 2011, Li et al., 2011, Toderici and Stamp, 2013]. Pour ce faire, différentes techniques sont utilisées, et qui sont présentées dans la suite de la section.

Le renommage des registres

La technique de renommage des registres (en anglais Registry Renaming) est une technique efficace et très simple à mettre en œuvre. Cette technique consiste à remplacer un registre utilisé dans une instruction par un autre, tel illustré dans la figure 3.8. Cette technique est efficace dans le sens où elle permet de changer les séquences de bits (binary pattern en anglais) du programme. Cependant, elle ne change pas la séquence des instructions (les Opcodes), ce qui la rend plus facile à détecter [Toderici and Stamp, 2013, Aycock, 2006, LeDoux and Lakhotia, 2015].

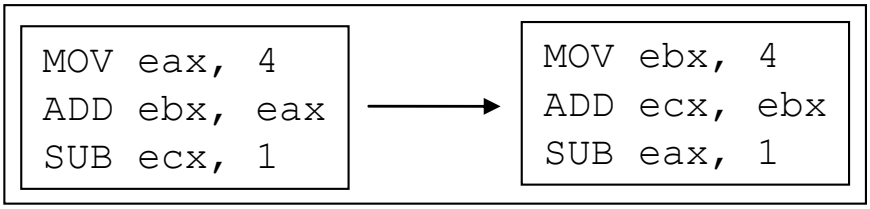

FIGURE 3.8 – Exemple d'un renomage de registre [Toderici and Stamp, 2013].

Remplacement des instructions équivalentes

Certains malwares métamorphiques sont capables de remplacer une partie de leurs instructions avec d'autres instructions équivalentes. Par exemple, `MOVE eax, 0`, peut être remplacée par `SUB eax, eax` ou encore `XOR eax, eax` [Toderici and Stamp, 2013]. La figure 3.9 illustre un cas d'équivalence d'instructions.

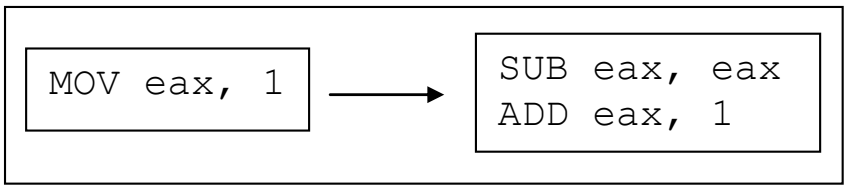

FIGURE 3.9 – Exemple d'un replacement d'instructions équivalentes [Toderici and Stamp, 2013].

Réarrangement d'instructions

Dans ce type de méthodes, l'ordre avec lequel les instructions apparaissent dans le code est modifié (figure 3.10), du moment que ces instructions ne dépendent pas du résultat de celles qui les précèdent [Aycock, 2006]. Quand les instructions sont réarrangées, leur signature est systématiquement rompue cependant l'exécution du programme ne sera pas affectée [Toderici and Stamp, 2013].

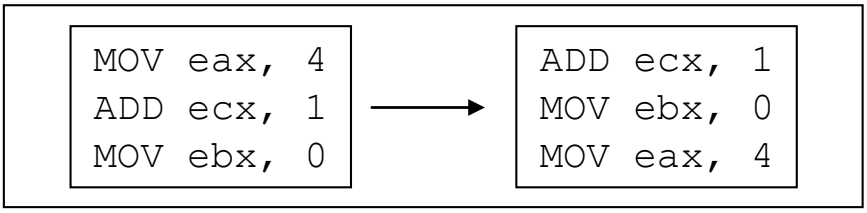

FIGURE 3.10 – Exemple d'un réarrangement d'instructions [Toderici and Stamp, 2013].

Insertion de code poubelle

Le code poubelle (ou code mort) est un code qui sera inséré lors du processus de mutation du malware mais qui ne sera jamais exécuté ou qu'il n'affectera pas l'exécution du programme, néanmoins il changera sa signature [Toderici and Stamp, 2013]. La figure 3.11 illustre un exemple d'insertion de code poubelle dans un malware.

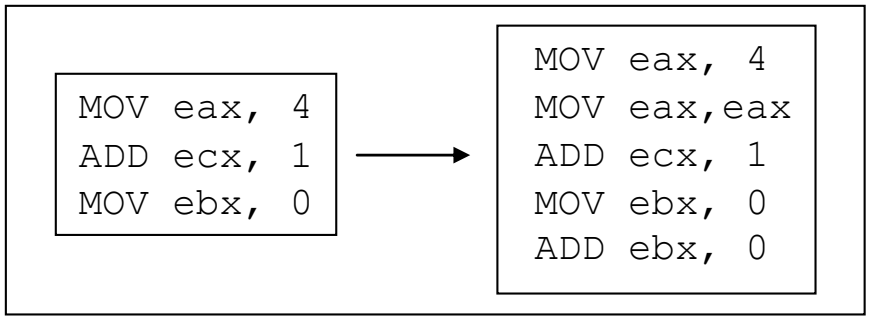

FIGURE 3.11 – Exemple d'insertion de code poubelle [Toderici and Stamp, 2013].

3.5 Techniques d'analyse et de détection des malwares

La détection de malwares est un problème de classification binaire, à savoir, décider si un code donné appartient à la classe des codes malicieux ou celle des codes bénins [Aycock, 2006, Wang et al., 2009a]. Le processus de détection d'un malware nécessite le passage par une phase d'analyse du code, celle-ci va permettre d'en extraire différents attributs, qui permettront la classification du fichier. On distingue deux types d'analyses, qui sont l'analyse statique et l'analyse dynamique [Siddiqui et al., 2008b, Galal et al., 2015]. L'analyse dynamique nécessite l'exécution du programme, cela est fait dans un environnement contrôlé, généralement construit à l'aide d'un émulateur (environnement virtuel) [Aycock, 2006]. L'analyse statique quant à elle, ne nécessite pas l'exécution du programme, et consiste à l'inspecter en procédant à son désassemblage. Le désassemblage où l'ingénierie inverse (en anglais reverse engineering) est la reconversion d'un programme, en code machine, au code assembleur. Les méthodes de détection de malwres quant à elles, peuvent être divisées en trois grandes catégories, qui sont les techniques basées signature, les techniques comportementales et les techniques heuristiques.

Les méthodes basées signature sont largement utilisées par les programmes antivirus commerciaux [Nazario, 2004, Filiol, 2009, Vinod et al., 2009, Shabtai et al., 2009]. Ce genre de méthodes repose sur la représentation de chaque malware en utilisant une signature. Une signature étant *une séquence d'octets qui est caractérisée par le fait qu'elle est unique pour*

chaque malware analysé [Ye et al., 2010].Celle-ci est générée à partir des séquences d'octets extraites de la totalité ou d'une partie du fichier analysé. À titre d'exemple, le malware Chernobyl cité précédemment possède la signature suivante : `E8000000005B 8D4B42515050 0F014C24FE5B 83C31CFA8B2B` [Toderici and Stamp, 2013]. Après leurs extractions, les signatures sont ensuite stockées dans ce qu'on appelle une base virale, et à chaque fois qu'un fichier est analysé, sa signature est générée et est comparées à toutes les autres présentes dans cette base. Ces techniques sont réputées pour être rapides et très précises pour la détection des malwares connus (malwares ayant leur signature dans la base virale). Cependant, elles sont incapables de détecter les malwares inconnus (Zero-Day), ainsi que les variantes d'un même malware (malwares obscurcis) [Ye et al., 2010, Sikorski and Honig, 2012]. En outre, Le processus de génération de la signature virale est très fastidieux et requiert des moyens techniques et humains importants. De ce fait, la mise à jour de la base virale peut prendre un certain temps et par conséquent le malware peut entre temps avoir causé des dégâts considérables avant que l'antivirus ne puisse le détecter [Aycock, 2006, June, 2010, Potter and Day, 2009, Shabtai et al., 2009]. C'est le cas par exemple, du ver SQL Slammer présenté précédemment qui a pu infecter plus de 90% des machines vulnérables en quelques minutes seulement [Jacob et al., 2008].

Les méthodes comportementales, consistent à construire un profil représentant soit un comportement normal, ainsi tout autre comportement déviant de ce dernier sera jugé comme étant malicieux. Soit en construisant un profil de comportement malicieux et ainsi sera jugé comme malicieux tout comportement similaire à ce dernier [Jacob et al., 2008]. Une approche comportementale passe par trois phases : la collecte d'information, la génération du modèle de comportement, et la phase de décision [Jacob et al., 2008]. La collecte d'informations se fait d'une manière statique ou dynamique, celle-ci consiste à collecter des informations relatives au comportement du malware. En effet, les actions qu'un programme accomplit (l'analyse dynamique) ou pouvant être accomplies (analyse statique du code machine) durant son exécution telles que l'activité réseau, l'accès aux fichiers, appels systèmes, etc, sont enregistrées dans des journaux. La génération du modèle comportemental consiste à analyser les données collectées durant la phase précédente afin d'en extraire celles jugées comme étant les plus pertinentes. Celles-ci seront regroupées afin de construire une signature (modèle de comportement malicieux), qui contrairement aux signatures classiques, celle-ci peut être utilisée pour identifier un ensemble de malwares au lieu d'un seul. La phase de décision s'effectue en essayant de trouver une correspondance entre la signature extraite à partir du fichier analysé et celles représentant les comportements malicieux établis au préalable. Ce type de méthodes est efficace pour détecter les malwares inconnus vu que la signature utilisée est générique est peut donc représenter des comportements malicieux qui peuvent être communs à une ou plusieurs

catégories de codes malicieux [Bazrafshan et al., 2013]. En outre, cette technique permet la détection des malwares obscurcis vu que ces derniers changent de forme mais conservent le même comportement.

Les méthodes heuristiques tentent de trouver des caractéristiques comportementales et/ou structurelles relatives aux fichiers malicieux et qui vont permettre leur distinction des fichiers bénins[Skoudis and Zeltser, 2004]. À leurs débuts, ces techniques étaient basées sur la notion de règles de détection. Celles-ci étaient construites par des experts humains et consistaient à assigner des poids aux fonctionnalités et/ou aux caractéristiques trouvées dans le fichier analysé. Dans le cas où la somme des poids dépassait un certain seuil, le fichier est considéré comme étant malicieux [Skoudis and Zeltser, 2004, Aycock, 2006]. Dans le but d'automatiser ce type de méthodes, les experts en sécurité informatique ont utilisé des techniques d'apprentissage automatique (en anglais machine learning) et plus précisément l'apprentissage supervisé afin de construire un modèle de classifieur permettant de distinguer les malwares des fichiers bénins [Bazrafshan et al., 2013, Shabtai et al., 2009]. C'est dans cette catégorie de techniques de détection que se situe notre travail, à savoir, la détection des malwares par les méthodes de machine learning basées sur l'analyse statique des codes PE.

Dans la section suivante, nous allons présenter les différents aspects relatifs a ce genre de méthodes de détection.

3.6 Méthodes de détection de malwares basées sur l'apprentissage automatique

Selon C.Ledoux et A.Lakhotia [LeDoux and Lakhotia, 2015], *l'apprentissage automatique se réfère au fait de doter un programme de la faculté d'apprendre un concept, et il est supervisé s'il adopte un apprentissage par l'exemple.* Dans notre cas, les exemples (aussi appelés données d'apprentissage) sont les programmes étiquetés, c.-à-d. dont on connait leur classe (malware ou bénin). Ces derniers seront représentés en utilisant certains de leurs attributs (caractéristiques). En se basant sur les attributs des programmes (données d'apprentissage) un modèle de classification est construit (phase d'apprentissage) et ce dernier sera évalué (phase de test) en utilisant un autre ensemble de programmes étiquetés appelés ensemble de test. La figure 3.12 illustre les phases d'apprentissage et de test d'un classfieur basé sur l'apprentissage automatique.

Les attributs extraits à partir des programmes analysés peuvent être en nombre élevé, et peuvent également contenir des attributs n'ayant aucune pertinence. De ce fait, ceci peut avoir une mauvaise influence sur les performances du système en matière de temps de traitement,

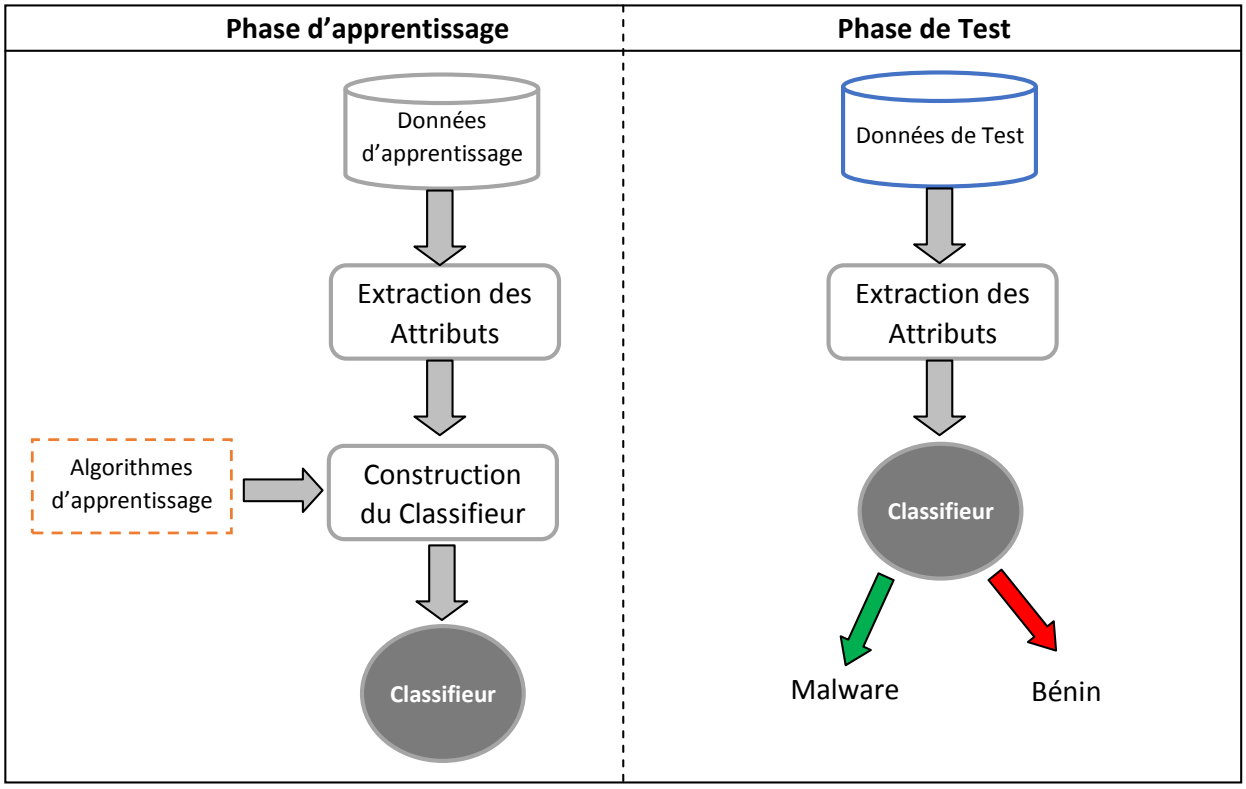

FIGURE 3.12 – Phases d'apprentissage, et de test d'un classifieur basé sur l'apprentissage supervisé [Shabtai et al., 2009].

et de précision. En effet, utiliser un nombre trop élevé d'attributs fera perdre au classifieur sa capacité de généralisation pour de nouvelles données à classifier (le cas de sur-apprentissage). Par conséquent, le système de détection doit être doté d'un mécanisme pour la sélection des attributs pertinents.

Dans ce qui suit, nous allons présenter les différents éléments composant un système de détection de malwares, basé sur l'apprentissage automatique supervisé (AAS), à savoir, le type d'attributs utilisés, la méthode de sélection des attributs, ainsi que l'algorithme de classification [Shabtai et al., 2009, Bazrafshan et al., 2013]. Cette décomposition est présentée dans la figure 3.13.

3.6.1 Les attributs utilisés

Consiste à représenter un fichier à l'aide d'un ensemble d'attributs le caractérisant. Dans cette thèse, nous allons nous intéresser à l'analyse des fichiers au format PE (Portable Executable), qui a été adopté par Microsoft pour le codage des fichiers exécutables sous l'environnement

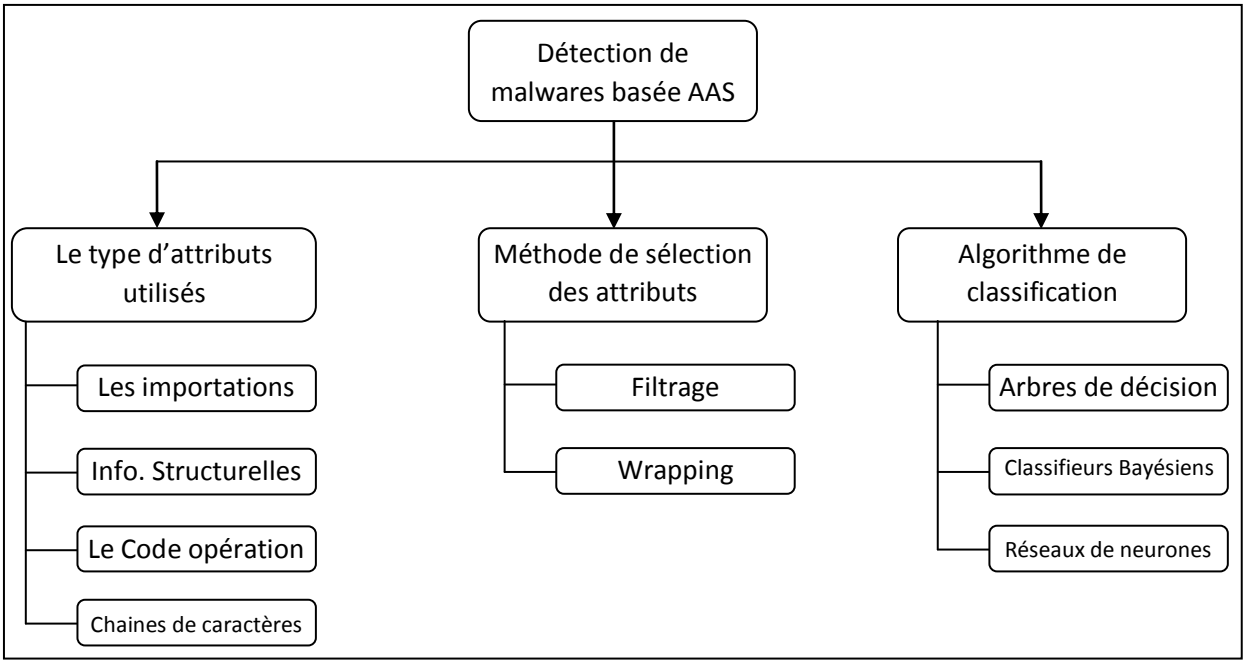

FIGURE 3.13 – Détection de malwares basée sur l'apprentissage automatique.

Windows [Pietrek, 1994] [5]. De ce fait, l'analyse statique de ce type de fichiers permet d'en extraire différents attributs qui sont principalement les informations structurelles du PE, les importations qui représentent les différentes DLLs utilisées (Dynamic Link Library, en français, bibliothèque de liens dynamiques) ainsi que les APIs (Application Programming Interface, en français interfaces de programmation d'applications) importée de chaque DLL, les instructions machine, et les chaines de caractères.

- **Les importations :** Presque tous les programmes utilisent des interfaces de programmation d'applications (API) afin d'interagir avec le système d'exploitation. Ces interfaces sont stockées au niveau de bibliothèques dédiées, à savoir, des DLLs. Lors de l'analyse statique du fichier exécutable les APIs sont extraites à l'aide de la Table des importations (IAT, en anglais Import Address Table). C'est la méthode adoptée dans notre cas, et celle-ci sera abordée dans le chapitre suivant.

- **Les informations structurelles du PE :** Elles sont composées des différentes informations extraites à partir du fichier PE analysé, telles que les informations des entêtes, les sections, etc. Ces attributs font également l'objet d'une partie de notre travail, et vont être présentés en détail dans le chapitre suivant.

5. À partir de la version 32 bits.

– **Le code opération :** Un code opération (Opcode) est une partie de l'instruction en langage machine qui identifie l'opération à exécuter. Plus précisément, un programme est défini comme une série d'instructions ordonnées en assembleur. Une instruction est une paire composée d'un code-opération et d'un opérande, ou une liste d'opérandes.

– **Les chaines de caractères :** représentent les séquences d'octets inclues dans le programme prouvant être interprétées comme une séquence de caractères ayant sémanti - quement un sens [LeDoux and Lakhotia, 2015]. Les chaines de caractères sont nécessaires à l'affichage de messages, la représentation des adresses WEB, à la copie d'un fichier, d'un emplacement à un autre, ainsi qu'aux noms de fichiers longs [Sikorski and Honig, 2012].

3.6.2 Méthode de sélection des attributs

Ces méthodes ont pour objectif de réduire le nombre des attributs, en ne gardant que ceux ayant une grande contribution dans le processus de catégorisation. Les méthodes de sélection d'attributs peuvent être classées en deux grandes catégories, qui sont les méthodes de filtrage, et les méthodes de Wrapping [Chouaib, 2011, Alazab et al., 2014, Belaoued et al., 2015].

– **Le filtrage :**les méthodes basées sur le filtrage sont indépendantes de l'algorithme d'apprentissage. Elles ont comme entrée, un ensemble d'attributs, et produisent en sortie, un sous-ensemble d'attributs pertinents en fonction des caractéristiques générales des données (figure 3.14). Différentes caractéristiques sont employées pour l'évaluation des attributs, telles que, la corrélation, l'information mutuelle, le critère de Ficher, etc [Chouaib, 2011].

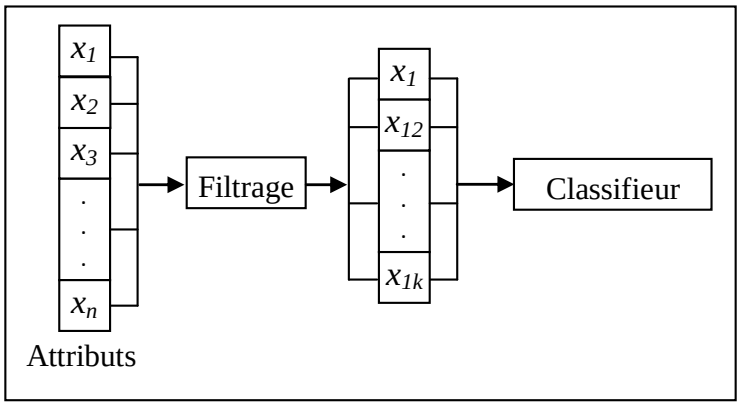

FIGURE 3.14 – Sélection d'attributs par filtrage [Chouaib, 2011].

– **Le Wrapping :** les techniques de Wrapping emploient des algorithmes de recherche heuristique (tels que le best first, Greedy Hill Climbing, etc.) afin de parcourir l'espace des attributs et en sélectionnant un sous-ensemble d'attributs qui sera directement évalué sur l'algorithme de classification employé. La figure 3.15 illustre le mécanisme de sélection d'attributs par la méthode du Wrapping [Chouaib, 2011].

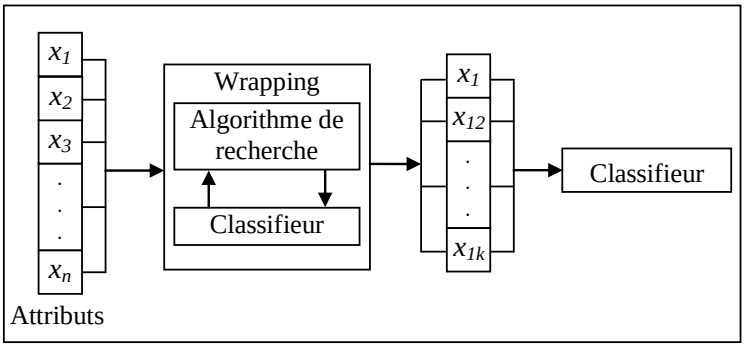

FIGURE 3.15 – Sélection d'attributs par Wrapping [Chouaib, 2011].

3.6.3 Algorithmes de classification

Il existe plusieurs catégories d'algorithmes de classification. Cependant, nous allons discuter ceux que nous jugeons êtres les plus utilisés dans le domaine de la détection des malwares, à savoir, les arbres de décision, les classifieurs bayésiens, ainsi que les réseaux de neurones.

– **Les arbres de décision :** Un arbre de décision est construit en se basant sur le principe des règles logiques, qui permettent de diviser l'ensemble des exemples d'apprentissage de manière hiérarchique [Fürnkranz et al., 2012]. En effet, chaque niveau de l'arborescence est construit en recherchant l'attribut le plus discriminant pour classifier un exemple. Les algorithmes à base d'arbres de décision peuvent avoir plusieurs variantes. Ils diffèrent généralement par la méthode de sélection de l'attribut discriminant. Par exemple, le cas de l'algorithme CART (Classification And Regression Tree, en français Arbre de classification et de régression) utilise le coefficient de Gini comme critère de sélection, l'algorithme C4.5 (J48) utilise la métrique du gain d'information. Ce dernier, sera présenté dans le chapitre suivant.

– **Les classifieurs Bayésiens :** Les classifieurs Bayésiens naïfs sont basés sur le théorème

de Bayes [Rish, 2001], qui est représenté par la formule suivante (Eq. 3.1) :

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{3.1}$$

Ce théorème est utilisé pour associer une probabilité (P) d'apparition d'un événement (A) sachant un autre événement (B). Il existe plusieurs algorithmes de classification dits bayésiens tels le classifieur Bayésien naïf, et les réseaux bayésiens [Shabtai et al., 2009]. Dans notre travail nous allons utiliser le classifieur Bayésien naïf, de ce fait, il sera revu avec un peu plus de détails dans le chapitre suivant.

– **Les réseaux de neurones artificiels :** Un réseau de neurones artificiels (RNA) [Singh and Chauhan, 2009] est un modèle de calcul qui est constitué d'un certain nombre d'unités de traitement (appelées neurones artificiels) qui communiquent en envoyant des signaux les uns aux autres sur un grand nombre de connexions pondérées. Ce type d'algorithmes s'inspirent du fonctionnement de la cellule nerveuse biologique (neurone). Un neurone artificiel est composé d'un ensemble d'entrées ainsi que leurs poids, une fonction d'activation, et enfin une fonction qui calcule la sortie du neurone. La phase d'apprentissage d'un RNA consiste à fournir des exemples d'entrées ainsi que leur sorties.

Comme mentionné précédemment, ces classifieurs sont entrainés en utilisant l'ensemble des résultats obtenus à partir de la phase de sélection des attributs (la phase d'apprentissage). Ensuite, le modèle de classifieur généré est évalué à l'aide d'un échantillon de test. Dans le chapitre suivant, nous allons présenter un ensemble de ces algorithmes et qui seront employés pour la réalisation de notre systèmes de détection de malwares.

3.7 Conclusion

Dans ce chapitre, nous avons présenté d'une manière globale les différents types de malwares, tout en décrivant les différents modes d'infection utilisés par chaque type. Nous avons aussi discuté des différentes méthodes d'obscurssissement employées par les hackers afin de rendre leurs malwares de plus en plus difficiles à détecter. Enfin, nous avons présenté les différentes techniques existantes pour la détection des malwares. Ceci nous permettra dans les chapitres suivants de proposer des méthodes de catégorisation de malwares, et plus loin dans ce manuscrit, une approche collective pour leur détection.

Chapitre 4

Approches Statistiques Décisionnelles pour la Détection des Malwares

Sommaire

4.1 Introduction

En sécurité informatique, quelle que soit la force d'un système de détection, du point de vue ingéneering, il est nécessaire qu'on soit en possession d'une puissante approche de décision pour catégoriser les menaces qui peuvent nuire à un système informatique. Dans cette optique, nous proposons dans ce chapitre deux approches statistiques pour la détection des

malwares PE, qui représentent en notre première contribution dans cette thèse. Dans la première approche, nous exploitons les informations extraites à partir du fichier PE, à savoir les informations contenues dans les entêtes du fichier ainsi que l'ensemble des APIs importées, et cela pour pouvoir décider si ce code est malicieux ou bénin. Nous avons utilisé le test du KHI^2 pour catégoriser si une information donnée du PE relève des malwares ou des codes bénins. Notre but étant d'aboutir à une catégorisation en temps réel des codes analysés. La deuxième approche, est basée sur une méthode d'analyse statistique de données, à savoir l'analyse des correspondances multiples (ACM) pour inférer les associations d'APIs Windows, qui peuvent se trouver dans les malwares. Ceci nous permet d'aboutir à une catégorisation multi-classes des malwares (virus, vers, chevaux de Troie, etc.).

4.2 Un système temps réel pour la détection des malwares PE

4.2.1 Prologue

Dans cette section, nous allons présenter notre approche pour la détection en temps réel des malwares PE. Celle-ci est basée sur l'analyse des APIs importées par les applications ainsi que l'ensemble des informations extraites à partir des entêtes du fichier PE (IPE). Notre approche repose sur l'observation que certaines APIs sont plus utilisées par les malwares que par les fichiers bénins, et inversement[Belaoued and Mazouzi, 2014, Belaoued and Mazouzi, 2016a]. En outre, certaines caractéristiques (altérations) dans les entêtes des fichiers PE ont été observées chez les malwares et non chez les applications bénignes[Belaoued and Mazouzi, 2015b]. Ceci doit être identifié par la différence dans les fréquences d'apparition des attributs dans les deux catégories d'application (Malawares et bénins). Pour pouvoir distinguer d'une manière précise, parmi ces attributs, lesquels ont une importance significative pour la distinction entre les malwares et les codes bénins, nous avons utilisé la méthode statistique du KHI^2 (CHI-DEUX) [Harrison, 2009]. Cette méthode qui utilise un test d'hypothèses, nous a permis de réduire considérablement le nombre des attributs n'ayant pas une forte corrélation avec l'une des deux catégories des PE. Après avoir éliminé les attributs n'ayant aucune contribution dans le processus de catégorisation, nous allons employer le coefficient Phi (ϕ), qui est généralement utilisé pour mesurer le degré de signification des attributs [Chedzoy, 2006]. Ce dernier va nous permettre de diviser les attributs en sous ensembles selon leurs degrés de signification. Enfin, un algorithme de classification est employé afin de permettre la catégorisation des fichiers (malware, bénin).

4.2.2 Le format PE

Le PE (Portable Executable) est un format de fichier commun aux exécutables et aux DLLs sous le système d'exploitation Windows à partir de la version 32 bits [Pietrek, 1994]. Un fichier PE est composé d'un entête MS-DOS, d'un entête PE, d'une table des sections, ainsi qu'un ensemble de sections. La structure d'un fichier PE est illustrée dans la figure 4.1 suivante :

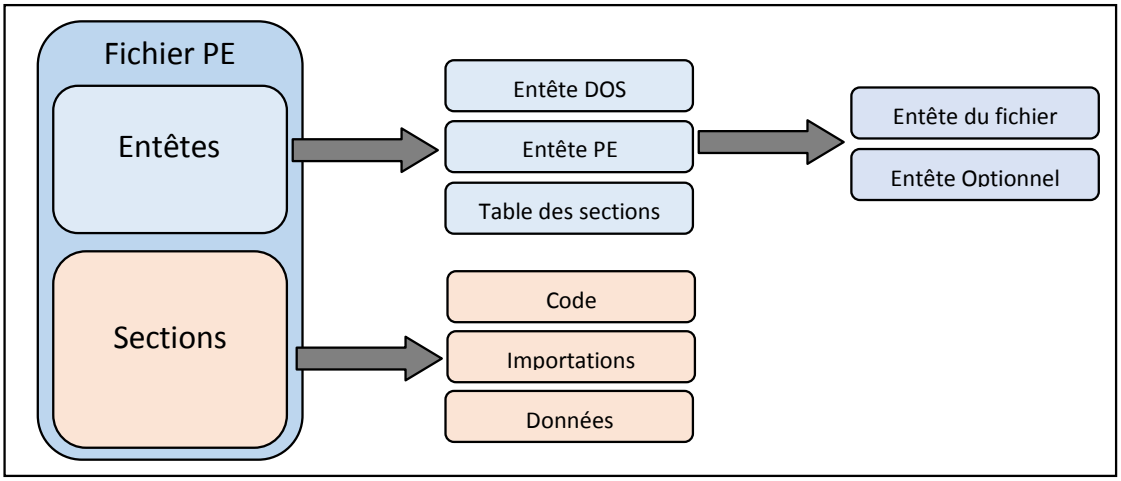

FIGURE 4.1 – Structure d'un fichier PE.

L'entête DOS

Situé au début du fichier PE, l'entête DOS est utilisé pour vérifier si le fichier est un exécutable valide ou non, cela dans le cas où il est exécuté à partir du système d'exploitation DOS (Disc Operating System).

L'entête PE

Il est composé de trois éléments qui sont : la signature, l'entête du fichier (File Header), et l'entête optionnel (Optional header).

- **Signature :** Elle permet d'identifier le fichier, est doit impérativement contenir la valeur `0x00004550`, soit $PE\backslash 0\backslash 0$ tel que $\backslash 0$ est un octet nul.
- **L'entête du fichier (File Header) :** est une structure de type `IMAGE_FILE_HEADER`. Elle contient des informations importantes relatives à la structuration du fichier (voir figure 4.2). Si l'on prend l'exemple du champ `Characteristics`, celui-ci fournit des informations sur le type du fichier (exécutable (.EXE) ou DLL). Le champ `Machine` quant à lui, spécifie le type du processeur pour lequel le fichier est destiné pour (Intel i860, Intel I386, etc.).

```
typedef struct _IMAGE_FILE_HEADER {
  WORD  Machine;
  WORD  NumberOfSections;
  DWORD TimeDateStamp;
  DWORD PointerToSymbolTable;
  DWORD NumberOfSymbols;
  WORD  SizeOfOptionalHeader;
  WORD  Characteristics;
}IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

FIGURE 4.2 – Structure du File-Header.

– **L'entête Optionnel (Optional Header) :** Contient des informations additionnelles à celles déjà fournis dans le File Header. Par exemple, les champs `MinorLinkerVersion` et `MinorOperatingSystemVersion` contiennent respectivement des informations sur la version du Linker utilisé pour créer les fichiers, et la version minimale du système d'exploitation requise pour utiliser le fichier. La figure 4.3 présente la structure de L'entête Optionnel.

```
typedef struct _IMAGE_OPTIONAL_HEADER {
WORD  Magic;
BYTE  MajorLinkerVersion;
BYTE  MinorLinkerVersion;
DWORD SizeOfCode;
DWORD SizeOfInitializedData;
DWORD SizeOfUninitializedData;
DWORD AddressOfEntryPoint;
…     …
IMAGE_DATA_DIRECTORY DataDirectory;
}IMAGE_OPTIONAL_HEADER, *PIMAGE_OPTIONAL_HEADER;
```

FIGURE 4.3 – Structure de l'entête optionnel.

L'entête optionnel contient également un champ nommé DataDirectory qui est un vecteur contenant 16 structures de type `IMAGE_DATA_DIRECTORY`. Chacune d'elles est liée à une structure de données importante. Une de ces structures est l'`IMAGE_DIRECTORY_ENTRY_IMPORT` qui contient des informations sur toutes les importations. Les importations sont l'ensemble des bibliothèques (DLL) dont l'exécutable est lié. Une grande partie des fonctionnalités de Windows se trouve dans les DLLs qui fournissent aux applications un moyen pour interagir avec les services de base de Windows [Eilam, 2011]. Quand un exécutable est chargé, le Windows Loader se charge de la lecture dans la structure du fichier PE et du chargement l'image de l'exécutable en mémoire, ainsi que toutes les DLLs utilisées. L'exécutable répertorie également

toutes les fonctions de chaque DLL dont il aura besoin. L'`IMAGE_DIRECTORY_ENTRY_IMPORT` pointe vers un vecteur de structures `IMAGE_IMPORT_DESCRIPTOR`. Chaque structure à une taille de 20 octets et contient des informations sur les DLL utilisées par le fichier PE pour appeler les APIs. Le nombre de structures `IMAGE_IMPORT_DESCRIPTOR` est égal au nombre de DLLs importées. Le champ `OriginalFirstThunk` de l'`IMAGE_IMPORT_DESCRIPTOR` contient un RVA (Relative Virtual Address) qui pointe vers l'IAT (la table des importations, pour Import Address Table). Celle-ci est un tableau de pointeurs de fonctions, rempli par le Windows Loader lors du chargement des DLLs. Lorsque l'application a été compilée, elle a été conçue de telle sorte que tous les appels d'APIs ne seront pas utilisés via des adresses codées en dur directement, mais plutôt par un pointeur de fonctions [Singh, 2009]. L'IAT est située dans la section des importations, qui est présentée ci-dessous.

La table des sections et les sections

La table des sections est située juste après l'entête PE. Il s'agit d'un tableau de structure de type `IMAGE_SECTION_HEADER`. Celle-ci contient les informations sur les sections devant être chargées. C'est le champ `NumberOfSections` qui indique le nombre d'entrées dans cette table. Un fichier PE contient différentes sections, chacune d'elles contient des informations sur des éléments différents, comme expliqué ci-dessous [Pietrek, 1994, Singh, 2009, Eilam, 2011] :

- La section du code (.txt) : contient l'ensemble des instructions d'un programme.
- La section des importations (.idata) : contient les informations relatives aux fonctions importées par le fichier.
- La section de données (.data) : Contient toutes les données initialisées (globales et statiques)
- La section des données (.rdata) : Contient les données en lecture seule tels que les chaines littérales (strings), ainsi que les constantes.
- La section des exportations (.edata) : contient des informations sur les fonctions exportées par le fichier.
- La section des ressources (.rsrc) : Contient des informations sur les menus et les boites de dialogues.
- La section de relocation (.reloc) : contient des informations sur la relocation lors du chargement de l'image.

4.2.3 Travaux similaires

Avant de présenter dans le reste de ce chapitre nos approches de catégorisation de malwwares, nous survolons dans cette section certains travaux bien cités dans la littérature, ayant exploité

le format PE afin de détecter des fichiers malveillants.

Schultz et al. [Schultz et al., 2001] ont mis en place le premier système de détection de malwares basé sur des techniques d'apprentissage automatique. Les auteurs ont étudié différentes informations contenues dans le fichier PE tels que des chaines de caractères, les APIs, et la séquences d'octets. Ils ont utilisé une méthode de classification basée sur un algorithme Bayésien naïf, et ils ont obtenu une précision globale de 97,11% en utilisant les chaines de caractères comme attributs.

La méthode proposée par C.Wang et al. [Wang et al., 2009b] utilise également le classifieur Bayésien naïf avec les appels d'APIs pour la classification des malwares. Les APIs extraites ont été utilisées pour construire des modèles de comportements suspects, en regroupant certaines APIs en se basant sur un ensemble de scénarios d'actions qu'un malware (de type virus) peut accomplir. Parmi les modèles de comportement que les auteurs ont utilisé, on trouve la recherche d'un fichier à infecter, l'écriture des données malveillantes dans le fichier, etc. Ils ont obtenu 93,7% de précision de détection.

PE-Miner [Shafiq et al., 2009] est présenté comme un système pour la détection en temps réel des malwares PE. Les auteurs ont pu extraire statiquement un ensemble composé de 198 caractéristiques structurelles du PE, telles que les informations d'en-tête, le nombre de sections que comporte le PE, etc. Les auteurs ont proposé une méthode basée sur le gain d'information pour sélectionner les caractéristiques les plus pertinentes. Ils ont évalué leur système en utilisant différents algorithmes de classification et ont été en mesure d'atteindre 99% de taux de détection et environ 0.5% comme taux de faux positifs. Le processus de catégorisation de PE-Miner prend en moyenne 0,244 secondes par fichier.

Ye et al. ont proposé CIMDS [Ye et al., 2010], qui est une amélioration de leur système de détection de malwares précédent, appelé IMDS (Intelligent Malware Detection Systeme) [Ye et al., 2008]. Comme son prédécesseur, CIMDS est basé sur l'analyse des séquences d'exécution des appels d'APIs. Ces APIs sont utilisées pour construire des règles d'associations [Shen et al., 2002, Witten et al., 2011], qui vont permettre la classification des malwares. Ils ont utilisé la méthode du Khi^2 comme méthode de réduction des attributs (éliminer les règles insignifiantes), comme méthode de classement des attributs (de la règle la plus significative à la moins significative), et comme méthode de sélection des attributs (les n meilleurs règles). Le système a pu atteindre 67,5% de précision et 88,16% de taux de détection, ce qui nécessite des améliorations. Cependant, il a un très bon temps de détection qui est de 0.09 secondes par fichier. Les auteurs dans [Ding et al., 2013] ont également présenté un système de détection de logiciels malveillants qui utilise une méthode de classification à base de règles d'associations construites à partir des séquences d'appels des APIs. Ils ont proposé une méthode de réduction et de sélection des attributs obtenus en ne gardant que les 1000

meilleurs règles et ce, en se basant sur deux critères, qui sont la fréquence de document (Document Frequency) et le gain d'information (Information Gain). Le système atteint 91,2% de précision et un taux de détection de 97,3%.

La méthode décrite dans [Shankarpani et al., 2012], est également basée sur l'analyse des APIs extraites à partir des fichiers PE analysés. Dans la phase d'apprentissage, les APIs qui sont extraites statiquement, sont utilisées pour générer des signatures pour les fichiers malicieux qui seront ensuite stockées dans la base de signatures. Afin de réduire le nombre des signatures d'APIs et ne conserver que les plus pertinentes d'entre elles, les auteurs ont utilisé un système de pondération basé sur la méthode TF-IDF (de l'anglais Term Frequency-Inverse Document Frequency). Les auteurs ont proposé deux méthodes pour la classification des malwares, la première consiste à comparer les appels d'APIs du fichier analysé à celles stockées dans la base des signatures en calculant leur degré de similarité en combinant trois métriques différentes, à savoir la similarité du cosinus, la mesure Jaccard, et la corrélation de Pearson. La seconde consiste en l'utilisation d'un algorithme de classification, à savoir, la machine à vecteur de support (Support Vector Machine, abrévié SVM). Les auteurs ont obtenu une précision globale de 91,5%.

Les auteurs dans [Toderici and Stamp, 2013] ont combiné le test du Khi^2 et un modèle de Markov caché (Hiddn Markov Model, abrévié HMM) pour détecter les logiciels malveillants en utilisant les séquences de codes opération (Opcode). Ils ont extrait les Opcodes à partir des fichiers analysés en utilisant un désassembleur tiers qui est IDA Pro. Les auteurs ont utilisé la méthode du Khi^2 pour identifier l'ensemble d'instructions qui sont susceptibles d'être utilisés par le générateur automatique de malwares NGVCK (Next Generation Virus Construction Kit) pour générer des variantes de logiciels malveillants. Ils ont ensuite fait l'apprentissage du HMM en utilisant le jeu d'instructions obtenu afin de catégoriser les fichiers. Ils ont expérimenté leur méthode sur 200 codes malveillants et 40 bénins. Le système proposé a une précision globale de 91%. L'inconvénient de cette méthode est l'utilisation du désassembleur IDA Pro. Celui-ci rend le système partiellement automatique. En outre, leur analyse a été limitée aux logiciels malveillants qui sont générés en utilisant NGVCK, ce qui est assez restrictif.

Les Auteurs dans [Salehi et al., 2014] ont proposé un système de détection de logiciels malveillants qui est basé sur l'analyse des fonctions APIs et leurs arguments. Ils ont utilisé une méthode d'extraction de caractéristiques basée sur l'analyse dynamique des fichiers en utilisant un environnement virtuel. Ils ont évalué leur méthode en utilisant différents classifieurs, et ils ont obtenu une précision globale de 98,1%. L'utilisation des arguments des APIs nécessite l'exécution du programme pendent 2 minutes. Par conséquent, cette méthode n'est pas adéquate pour un déploiement en temps réel.

En considérant les travaux cités en haut, et d'autres dont nous n'avons pas jugé nécessaire d'encombrer avec, le lecteur, nous pouvons constater que l'analyse des fichiers exécutables (au format PE) est actuellement intensivement utilisé pour la catégorisation des codes exécutables. Certaines approches sont basées sur des techniques inspirées du data mining et ne font aucune hypothèse sur le comportement du code quand il est exécuté. Certains auteurs modélisent le comportement d'une manière statique, et raisonnent dessus pour décider si le comportement modélisé dévoile une activité malveillante ou non. Une autre catégorie de travaux, procède selon une approche dynamique pour dévoiler des aspects jugés opportuns pour la détection de malwares. Pour se faire, les règles de décision peuvent être basées sur des heuristiques, ou bien sur des attributs calculés en utilisant des méthodes relavant du data mining.

4.2.4 Approche proposée pour la détection en temps réel des malwares PE

Comme illustré dans la figure 4.4, notre approche pour la détection des malwares passe par trois phases, qui sont : l'extraction et le pré-traitement des attributs, la sélection des attributs, et enfin, la phase de décision.

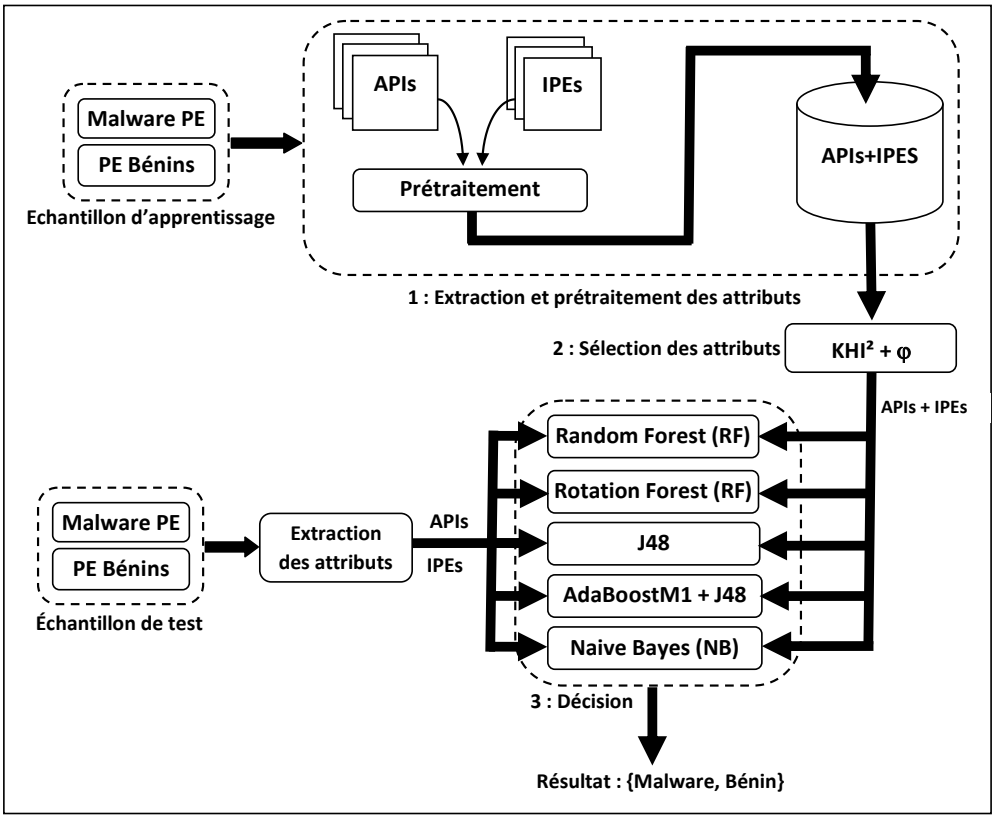

FIGURE 4.4 – Architecture du système proposé pour la détection des malwares PE.

Phase d'extraction et pré-traitement des attributs

Notre système de catégorisation de codes PE repose sur l'analyse de deux types d'attributs, qui sont les appels d'APIs et les informations stockées dans les champs de l'en-têtes PE (IPEs). Pour extraire ces deux types d'attributs à partir d'un fichier PE et calculer leurs fréquences, nous avons développé un module sous Python. La méthode d'extraction utilisée est basée sur une analyse statique de l'IAT pour les APIs et l'en-tête optionnel du PE pour les IPEs. Les figures (figure 4.5, figure 4.6), présentent un aperçu de quelques lignes de code Python que nous avons écrit pour extraire les appels d'APIs et les IPEs à partir d'un ensemble de fichiers PE contenus dans un dossier nommé 'Malware_Samples'.

```
1   import pefile
2   import os
3   API_list = []
4   PATH = "C:\Malware_Samples"
5   for FILE in os.listdir(PATH)
6    current_file =os.path.join(PATH, FILE)
7    pe= pefile.PE(current_file)
8    for entry in pe.DIRECTORY_ENTRY_IMPORT:
9       for API in entry.imports:
10       API_LIST.append(API.name)
11  ...
```

FIGURE 4.5 – Portion du script python pour l'extraction des APIs des fichiers PE.

```
1   import pefile
2   import os
3   TPF_list = []
4   PATH = "C:\Malware_Samples"
5   for FILE in os.listdir(PATH)
6    current_file =os.path.join(PATH, FILE)
7    pe= pefile.PE(current_file)
8    TPF_LIST.append("CheckSum"+str(pe.OPTIONAL_HEADER.CheckSum))
9    TPF_LIST.append("LoaderFlags"+str(pe.OPTIONAL_HEADER.LoaderFlags))
10  ...
```

FIGURE 4.6 – Portion du script python pour l'extraction des IPEs des fichiers PE.

Comme c'est présenté dans les lignes 8 et 9 du code source dans la figure 4.6, les IPEs sont représentés par la concaténation du nom du champ et de sa valeur. Par exemple, l'IPE `CheckSum0` signifie que le champ Checksum à une valeur égale a zéro.

Après l'extraction des APIs et des IPEs, on procède à la phase de pré-traitement qui consiste à enlever les APIs en doubles dans le même fichier PE, ainsi que le calcul de leurs fréquences d'appels dans les PE malveillants et les PE bénins. Les fréquences d'apparition des IPEs sont également calculées. À la fin de cette phase, nous obtenons une table composée de trois colonnes, qui représentent respectivement le nom de l'attribut, sa fréquence dans les

PE malveillants et sa fréquence dans les PE bénins, et un certain nombre de lignes, dont le nombre est égal au nombre d'attributs obtenus (APIs + IPEs).

Phase de sélection d'attributs

La phase de sélection d'attributs est selon notre approche, la deuxième phase dans le processus de catégorisation des malwares. Cette dernière, vise à sélectionner les attributs les plus pertinents à partir de la liste d'APIs et d'IPEs, obtenues précédemment. Nous avons utilisé pour cela une méthode statistique, à savoir, le test d'hypothèse du Khi^2 [Harrison, 2009].

Cette méthode est utilisée pour décider s'il existe une association significative entre deux variables qualitatives (corrélation). Cette association est exprimée par la distance D entre une fréquence observée O et une fréquence espérée attendu E (E représente le cas d'une parfaite indépendance entre les variables). Par conséquent, la force de corrélation entre les deux variables est proportionnelle à la distance D.

Dans notre cas, nous étudierons cette association entre deux variables : d'abord, la variable '**Attribut**' elle possède deux modalités : '**présent**' et '**absent**'. Cette variable représente la présence ou non d'un attribut (API ou IPE) dans un fichier PE. Ensuite, la variable '**PE-cat**', qui a également deux modalités : '**Malware**' et '**bénin**'. Cette variable représente les deux catégories qu'un fichier PE peut avoir.

Lors de la réalisation d'un test du Khi^2, nous commençons par définir les deux hypothèses H_0 et H_1 dont l'une sera acceptée, et l'autre rejetée. H_0 (hypothèse nulle) représente le cas d'interdépendance entre les deux variables. H_1 (hypothèse alternative) représente le cas de dépendance entre les deux variables. Dans notre cas, H_0 et H_1 sont définies comme suit :

- H_0 : la présence ou l'absence d'un attribut (API ou IPE) est indépendant du type de fichier PE (Malware ou Bénin).
- H_1 : la présence ou l'absence d'un attribut (API ou IPE) est dépendant du type de fichier PE (Malware ou Bénin).

Pour chaque attribut A, nous pouvons établir une table de contingence comme présenté dans le tableau 4.1.

	Attribut : présent	Attribut : absent	Total ligne
PE-cat : Malware	M_1	M_2	M
PE-cat : Benin	N_1	N_2	N
Total colonne	$M_1 + N_1$	$M_2 + N_2$	T

Tableau 4.1 – La Table de contingence d'un attribut A (API, ou IPE).

Les variables présentées dans le tableau 4.1 sont définies comme suit :

- **M** est **N** représentent respectivement le nombre total des PE malwares et des PE bénins.
- **T** est le nombre total des fichiers PE **(T=M+N)**.
- **M1** est le nombre de malwares PE qui contiennent A, et **M2** est le nombre de PE malwares qui ne contiennent pas A, tel que : **M = M1+M2**.
- **N1** est le nombre des PE bénins qui contiennent A, et **N2** est le nombre des PE bénins qui ne contiennent pas A, tel que : **N = N1+N2**.

En se basant sur la table de contingence, la valeur du KHI^2, annoté D^2 est calculée à l'aide de l'équation 4.1 :

$$D^2 = \sum_{r,c} \frac{(O_{r,c} - E_{r,c})^2}{E_{r,c}} \tag{4.1}$$

Où, $O_{r,c}$ représente la fréquence observée au niveau de la ligne r et la colonne c. $E_{r,c}$ est la fréquence espérée et qui est définie par l'équation 4.2, ci-après :

$$E_{r,c} = \frac{n_r \times n_c}{T} \tag{4.2}$$

Où n_r et n_c sont respectivement la somme sur la ligne r et la somme sur la colonne c.

Après avoir calculé les valeurs du Khi^2 pour les attributs extraits, nous devons déterminer laquelle des deux hypothèses H_0 ou H_1 est acceptée et laquelle est rejetée pour chaque attribut. Pour ce faire, nous devons comparer la valeur du Khi^2 (D^2) obtenue pour chaque attribut à un seuil, qui représente la valeur de Khi^2 théorique (χ^2), où H_0 est acceptée (H_1 rejetée) pour tout attribut qui a $D^2 \leq \chi^2$. Notez que selon le test d'hypothèse du Khi^2, les attributs pour lesquels H_0 est acceptée sont considérés comme non pertinents et seront systématiquement supprimés. La valeur théorique du χ^2 est obtenue en calculant d'abord le degré de liberté (DL), et en choisissant un niveau de signification α qui représente la probabilité de rejet d'une hypothèse, même si elle est vraie. En considérant DL et α, la valeur de χ^2 est obtenue à partir de la table de distribution du Khi^2 [Hald et al., 1952]. DL est calculé comme suit :

$$DL = (r-1) \times (c-1) \tag{4.3}$$

Où r et c représentent respectivement les nombres de modalités de la première et de la deuxième variable.

Après avoir retiré tous les attributs qui ne sont pas corrélés, et qui correspondent au cas où H_0 est acceptée, nous calculons le coefficient de corrélation ϕ pour les attributs restants. Le coefficient ϕ [Chedzoy, 2006] est une normalisation de la valeur du $Khi^2(D^2)$, qui ne peut être appliquée qu'aux tables de contingence de taille 2x2 (deux variables avec deux modalités). Le coefficient ϕ est utilisé pour mesurer le degré de dépendance, entre les deux variables [Farrington and Loeber, 1989]. Dans notre travail, ce coefficient est utilisé afin de grouper les attributs en sous-ensembles en fonction de leur force de corrélation (pertinence). ϕ est calculé comme suit :

$$\phi = \sqrt{\frac{D^2}{T}} \tag{4.4}$$

Où, T représente le nombre total des fichiers PE utilisés.

La valeur de ϕ varie entre 0 et 1, et la pertinence d'un attribut (API, ou IPE) est prportionnelle à cette valeur. Par conséquent, nous avons choisi d'utiliser quatre sous-ensembles qui contiendrons les attributs ayant respectivement $\phi > 0$ (représente le cas où $Khi^2 > 3.84$), $\phi \geq 0.25$, $\phi \geq 0.5$, et $\phi \geq 0.75$. Notre objectif est d'être en mesure d'identifier le nombre optimal d'attributs requis pour avoir la plus grande précision, et qui aidera aussi à réduire le temps de détection.

Phase de décision

Après avoir généré les différents sous-ensembles d'attributs, nous devons identifier lesquels des APIs ou des IPEs ou leurs combinaisons permettront d'obtenir les meilleures performances (à savoir, une précision élevée et un faible temps de détection). Par conséquent, nous avons d'abord évalué notre système à l'aide des sous-ensembles d'IPEs, puis en utilisant les sous-ensembles d'APIs, et enfin nous avons évalué notre système en essayant toutes les combinaisons possibles des sous-ensembles IPE-API. Nous avons utilisé différents algorithmes de classification disponibles dans WEKA, qui sont :

- **L'arbre de décision J48 :** Le J48 est une implémentation de l'arbre de décision C4.5 proposé par Ross Quinlan, et qui est lui-même une amélioration de l'algorithme ID3 [Witten et al., 2011]. Le classifieur J48 consiste à construire un arbre de décision en se basant sur les valeurs des attributs donnés comme échantillons d'apprentissage. Cela est fait en sélectionnant à chaque niveau de l'arbre les attributs pouvant mieux différentier l'ensemble des individus à l'aide de la métrique du gain d'information. Ces attributs qui représentent les feuilles de l'arbre sont étiquetés par un test qui va évaluer la valeur de l'attribut, ou dans le cas de nœuds finaux, où ceux-ci vont contenir les différentes

classes. Les branches de l'arbre sont étiquetés par la réponse au test du nœud qui leur est associé. La figure 4.7 illustre un exemple d'un arbre de décision J48, généré pour un ensemble d'IPEs.

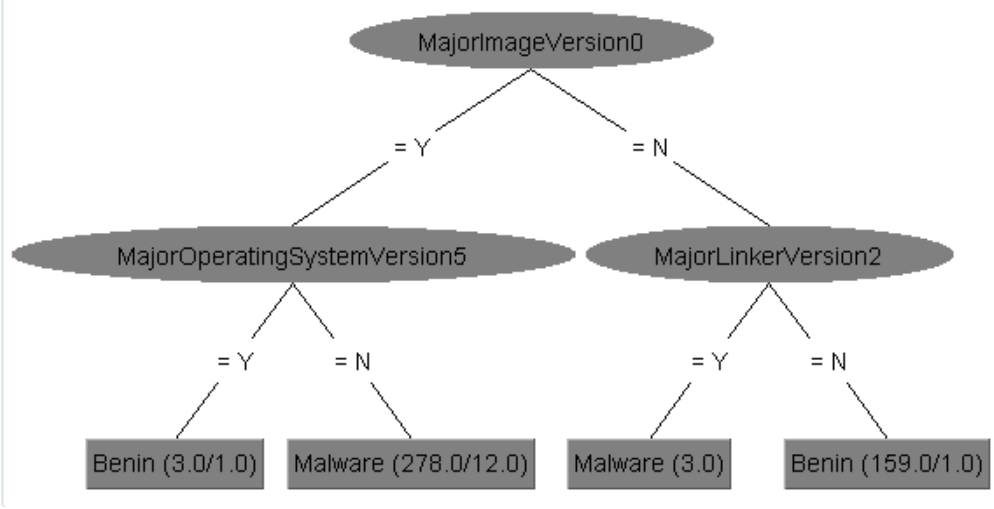

FIGURE 4.7 – Modèle de classifieur J48 généré à partir d'un ensemble d'IPEs.

– **Random Forest (Ran-F) :** L'algorithme Random Forest (forêt aléatoire décisionnelle) [Breiman, 2001], se compose d'un ensemble d'arbres de décisions. Ces derniers sont entrainés par le ré-échantillonage de l'ensemble d'apprentissage. Les différents classifieurs générés vont ensuite procéder à la classification (étiquetage) des données de test. Ainsi, la sortie du Random Forest est décidée par les votes exprimés par tous les arbres individuels [Dua and Du, 2011].

– **Rotation Forest (Rot-F) :** L'algorithme Rotation Forest [Rodriguez et al., 2006] (en français, rotation de forêts), est une méthode d'apprentissage qui consiste à combiner le résultat de classification de différents arbres de décision. La différence avec l'algorithme Random Forest réside dans le fait, que l'algorithme Rotation Forest procède à l'entrainement des arbres de décision, en utilisant la totalité de l'échantillon d'apprentissage [Kuncheva and Rodríguez, 2007]. En effet, pour chaque arbre de décision, les attributs en entrée sont divisés de manière aléatoire en k sous-ensembles, et cela en utilisant la méthode d'analyse en composantes principales (ACP). Ces arbres sont ensuite entrainés et les instances inconnues classifiées, et la classe finale est assignée à l'instance ayant le degré de confiance le plus élevé, parmi les résultats obtenus à partir des différents arbres décisions [Du et al., 2015].

– **Classificateur Bayésien naïf (NB) :** un classifieurs bayésien naïf affecte la classe la plus probable C_i à un exemple donné p décrit par son vecteur d'attributs X [Rish, 2001]. L'apprentissage de ce classifieur part de l'hypothèse que les attributs sont statistiquement indépendants de la classe donnée. De ce fait, cette probabilité qui est définie par $P(X \mid C_i)$ peut être estimée par la formule suivante :

$$P(X \mid C) = \prod_{i=1}^{n} P(X_i \mid C). \tag{4.5}$$

Le classificateur Bayésian naïf est remarquablement efficace dans la pratique, souvent en concurrence avec des techniques beaucoup plus sophistiquées [Rish, 2001].

– **l'algorithme AdaBoostM1 :** AdaBoost (pour Adaptative Boosting) fait partie de la catégorie des algorithmes de Boosting. Ces derniers, sont utilisés pour stimuler un algorithme d'apprentissage afin d'augmenter sa précision [Dua and Du, 2011]. Cela est fait en procédant à l'apprentissage répétitif d'un même algorithme en utilisant l'ensemble de données d'apprentissage que les modèles de classifieurs précédents n'ont pas pu correctement classifier [Dua and Du, 2011, Natani and Vidyarthi, 2013].

Pour notre cas, le module de décision prend en entrée les sous-ensembles d'attributs générés par le module de sélection des attributs, et l'ensemble des attributs extraits du fichier analysé. Les deux sont représentés sous forme de fichiers de données WEKA (fichier ARFF, voir section 4.2.5) [Witten et al., 2011], qui sont générés automatiquement en utilisant un script Python.

4.2.5 Implémentation

Dans ce qui suit, nous allons présenter les différents outils logiciels, qui nous ont permis d'implémenter notre système de détection de malwares.

Le langage Python

Python est un langage de programmation de Scripting orienté-objet [Lutz, 2013], qui a été créer par Guido van Rossum en 1989. Depuis ce temps, Python a connu des dizaines de versions et aujourd'hui, il est devenu un langage de programmation très robuste et stable. Python possède une large communauté de développeurs qui contribuent activement au développement des différents modules qui lui ont permis de devenir en peu de temps un langage polyvalent utilisé dans plusieurs domaines.

En plus de sa large utilisation par les particuliers et les établissements éducatifs, Python est utilisé par de grandes entreprises pour la réalisation de leurs produits, tels que Google dans son moteur de recherche, le système de partage des vidéos de YouTube repose grandement sur Python, et aussi BitTorrent, Intel, Cisco, HP, ... etc [Lutz, 2013].

Afin d'implémenter notre système, nous avons eu recours à un certain nombre de modules python tels que :

- **Le module PEfile :** Ce module Python multiplateforme permet d'analyser des fichiers Portable Executable (PE) [1]. La plupart des informations contenues dans les en-têtes PE sont accessibles, ainsi que les détails de toutes les sections et de leurs données. Les structures définies dans les fichiers d'en-tête de Windows seront accessibles comme attributs dans l'instance PE. Le nommage des champs respecte le schéma de nommage dans les entêtes. Le module offre une multitude de fonctionnalités, telles que l'inspection des différents entêtes, l'analyse des données des sections, la récupération des données intégrées, etc.

- **Le module OS :** Ce module [2] offre une panoplie de fonctionnalités dépendantes du système d'exploitation, telles que la manipulation des fichiers (lecture, écriture, copie, etc), la création des fichiers et répertoires temporaires, ainsi que La manipulation des chemins.

- **Le module Collections :** Ce module [3] fournit différentes structures de données de type conteneurs. Il nous a été d'une grande utilité lors de l'implémentation de notre systèmes en nous offrant un ensemble de fonctionnalités qui nous ont permis de manipuler aisément et avec une grande rapidité les différentes listes contenant les attributs extraits.

- **le module Math :** Ce module [4] fourni une grande variété de fonctions mathématiques de base, telles que la racine carré, le cosinus, etc.

1. Disponnible à : `https://code.google.com/p/pefile/`
2. Disponnible à : `https://docs.python.org/2/library/os.html`
3. Disponnible à : `https://docs.python.org/2/library/collections.html`
4. Disponnible à : `https://docs.python.org/2/library/math.html`

La plateforme WEKA

Weka[5] (pour Waikato Environment for Knowledge Analysis) [Witten et al., 2011] est un environnement qui regroupe une collection d'algorithmes pour la fouille de données (Data Mining en anglais), tels que les algorithmes de classification, de régression, etc. Weka permet soit d'appliquer une méthode d'apprentissage à un ensemble de données et d'analyser sa sortie, soit d'utiliser des modèles entrainés à générer des prévisions sur les nouvelles instances, ou bien d'appliquer plusieurs algorithmes d'apprentissages différents et de comparer leurs performances afin d'en choisir un pour la prédiction (nous nous situons dans ce cas là).

Les données traitées par Weka sont représentées sous la forme d'une table relationnelle répondant au format ARFF, Un exemple du format de fichier de données de WEKA (.arff) est présenté dans la figure 4.8.

```
1  @relation Malware
2
3  @attribute Class {Malware, Benin}
4  @attribute CheckSum0 {Y, N}
5  @attribute MajorImageVersion0 {Y, N}
6  @attribute setendoffile {Y, N}
7  @attribute getcommandlinew {Y, N}
8
9  @Data
10
11 Malware,N,N,N,N
12 Malware,N,N,N,N
13 Malware,N,N,N,N
14 Benin,N,N,N,N,N
15 Benin,N,N,N,N,N
16 Benin,N,N,N,N,N
```

FIGURE 4.8 – Un exemple d'un fichier ARFF contenant quatre attributs et six individus.

4.2.6 Expérimentation

Dans cette section, nous allons présenter les différents résultats expérimentaux, en commençant par la phase d'extraction des attributs et en terminant avec les performances de notre système.

5. Disponible à :`www.cs.waikato.ac.nz/ml/weka/`

Échantillons

À des fins d'expérimentation, nous avons recueilli un échantillon composé de 552 fichiers PE (338 logiciels malveillants et 214 programmes bénins), qui sera divisé en 80% comme ensemble d'apprentissage et 20% comme ensemble de test. Les fichiers PE infectés ont été obtenus à partir de la collection VxHeavens [6] et contiennent douze catégories de logiciels malveillants, comme indiqué dans le tableau 4.2.

Type de malware	**Nombre**
Backdoor	27
Email-Worm	19
Exploit	28
Hacktool	22
Network-Worm	16
P2P-Worm	17
Trojan	59
Trojan-Downloader	24
Trojan-Dropper	32
Trojan-Spy	18
Virus	42
Worm	34
Total	338

Tableau 4.2 – Echantillon de malwares utilisé.

Les fichiers PE bénins comprennent un ensemble de d'applications utilitaires qui ont été téléchargées à partir de Softpedia [7], et aussi quelques fichiers systèmes collectés à partir d'une installation récente de Windows XP. Dans notre travail, nous ne considérons que des programmes non compressés. Par conséquent, nous avons analysé nos échantillons en utilisant des outils connus de détection des Packers, tels que PEID [8], et ProtectionID [9]. Ces outils sont capables de détecter une grande variété de Packers, y compris les plus populaires tels qu'UPX, ASPack, et PECompact. Notez que les fichiers PE compressés sont la seule catégorie qui a été exclu de nos échantillons. Nous avons également analysé tous les fichiers à l'aide de plus de 40 AV différents disponibles au niveau du site web VirusTotal [10]. Plus de 30 AV ont identifié les fichiers PE infectés utilisés comme des logiciels malveillants, et aucun fichier bénin n'a été identifié comme malicieux.

6. Disponible à : `http://vxheaven.org/vl.php`
7. Disponible à :`http://www.softpedia.com/`
8. Disponible à :`https://www.aldeid.com/wiki/PEiD`
9. Disponible à :`http://pid.gamecopyworld.com/`
10. Disponible à :`https://www.virustotal.com/`

Résultats

Dans cette sous-section, nous présentons les résultats expérimentaux obtenus et cela en partant de la phase d'extraction des attributs jusqu'à la phase de décision. Après l'extraction et le pré-traitement des attributs, nous avons obtenu les résultats présentés dans les tableaux 4.3, et 4.4. Nous calculons ensuite les valeurs du Khi^2 pour les APIs et les IPEs obtenues, et on va retirer celles considérées comme non pertinentes, qui ont un $Khi^2 \leq 3,84$ (3,84 est la valeur χ^2 pour DL = 1 et α = 0,05). Ainsi, H_0 est rejetée et H_1 est acceptée pour 681 APIs et 50 IPEs, et H_0 est acceptée et H_1 rejetée pour 959 APIs et 540 IPEs, comme présenté dans les tableaux 4.5, et 4.6.

No.	**IPE**	**Fréq. malware (271)**	**Fréq. bénins (172)**
1	BaseOfCode4096	271 (100%)	172 (100%)
2	BaseOfData102400	4 (1%)	1 (1%)
3	BaseOfData106496	3 (1%)	1 (1%)
4	BaseOfData110592	1 (1%)	0 (0%)
5	BaseOfData118784	2 (1%)	0 (0%)
...	...	...	...
588	SizeOfUninitializedData95744	1 (1%)	0 (0%)
589	Subsystem2	226 (83%)	106(62%)
590	Subsystem3	45 (17%)	66(38%)

Tableau 4.3 – Aperçu des IPEs extraites.

No.	**API**	**Fréq. malware (271)**	**Fréq. bénins (172)**
1	Abort	4 (4%)	1 (1%)
2	Accept	21 (8%)	0 (0%)
3	Ace_Cleanup_Destroyer	1 (1%)	0 (0%)
4	ActivateKeyboardLayout	63 (23%)	4 (2%)
5	AddAccessAllowedAce	1 (1%)	18 (10%)
...	...	...	...
1638	xml_setuserdata	1(0%)	0(0%)
1639	zwquerysysteminformtaion	1(0%)	0(0%)
1640	zwunmapviewofsection	1 (1%)	0 (0%)

Tableau 4.4 – Aperçu des APIs extraites.

No.	**IPE**	Khi^2	ϕ
1	CheckSum0	2375.21	0.92
2	MajorImageVersion0	370.57	0.91
3	DllCharacteristics0	355.91	0.9
4	MajorOperatingSystemVersion5	346.02	0.88
5	MinorOperatingSystemVersion0	341.92	0.88
...	...	...	...
50	SizeOfInitializedData28672	3.86	0.09

Tableau 4.5 – Aperçu des IPEs retenues ($KHI^2 > 3.84$).

No.	**IPE**	Khi^2	ϕ
1	_p_commode	254.59	0.76
2	_setusermatherr	254.59	0.76
3	_exit	254.59	0.76
4	_xcptfilter	254.59	0.76
5	_controlfp	245.57	0.74
...	...	...	...
681	GetCurrentThread	3.85	0.09

Tableau 4.6 – Aperçu des APIs retenues ($KHI^2 > 3.84$).

Comme c'est montré précédemment, nous avons obtenu une liste finale de 681 APIs et 50 IPEs avec leurs valeurs du Khi^2 et du coefficient ϕ correspondant, comme indiqué dans les tableaux 4.5, et 4.6. Nous allons diviser ces attributs en différents sous-ensembles en fonction de leurs valeurs du coefficient ϕ. À la fin de la phase de sélection d'attributs, nous avons obtenu quatre sous-ensembles pour les APIs, qui sont A1, A2, A3, et A4 et quatre sous-ensembles pour les IPEs, qui sont H1, H2, H3, et H4. Ces derniers correspondent respectivement à quatre valeurs du ϕ, qui sont $\phi \geq 0.75$, $\phi \geq 0.5$, $\phi \geq 0.25$, et $\phi > 0$. Nous avons obtenu respectivement 5, 31, 297, et 681 APIs dans A1, A2, A3, et A4. Nous avons également obtenu 11, 14, 22, et 50 IPEs dans H1, H2, H3, et H4. Nous avons utilisé un sous-ensemble additionnel pour les API (A5) ainsi que pour les IPEs (H5) qui contiennent tous les attributs ayant été extraits (1640 APIs et 590 IPEs). Le but de l'utilisation de ces deux sous-ensembles supplémentaires (A5 et H5) est de vérifier si la méthode de sélection d'attributs proposée a amélioré les performances de notre système en matière de précision et de temps de détection.

Dans la prochaine section, nous allons évaluer la performance de notre système de détection en utilisant différents algorithmes de classification, qui seront entrainés avec les sous-ensembles d'attributs obtenus, ensuite, ils seront testés avec un échantillon de test. Le but étant de voir quel sous-ensemble ou combinaison de sous-ensembles va générer les meilleurs résultats.

Évaluation des résultats

Notre expérimentation a été menée sur une machine dotée des caractéristiques suivantes : Système d'exploitation Windows 7, 64 bits, processeur I3-2350 M 2,30 GHz et une RAM de 4 Go. Les phases d'extraction, de pré-traitement, ainsi que la sélection des attributs ont été mises en œuvre sous Python 27. Le module de décision a été mise en œuvre dans WEKA 3.7. La performance d'un système de détection de logiciels malveillants est généralement évaluée selon trois mesures différentes, comme on le verra ci-dessous :

- **Le taux de détection (TD) :** représente le pourcentage de malwares détectés parmi tous les programmes malveillants de l'ensemble de test, et il est calculé en utilisant l'équation 4.6 ci-dessous :

$$TD = \frac{\text{Nombre de Malwares Détéctés}}{\text{Nombre Total de Malwares}} \tag{4.6}$$

- **Le taux de faux positif (TFP) :** représente le pourcentage de fichiers bénins classés à tort comme logiciels malveillants parmi tous les fichiers bénins de l'ensemble de test, et il est calculé en utilisant l'équation 4.7.

$$TFP = \frac{\text{Nombre de Fichiers Bénins Classés Comme Malwares}}{\text{Nombre Total des Fichiers Bénins}} \tag{4.7}$$

- **La précision (PR) :** représente le taux de fichiers qui ont été classés correctement dans leur classe, et il est calculé en utilisant l'équation 4.8.

$$PR = \frac{\text{Nombre de Fichiers Correctement Classifiés}}{\text{Nombre Total de Fichiers}} \tag{4.8}$$

Vu que nous voulons aboutir à une détection en temps réel des logiciels malveillants, nous devons prendre en considération le temps de détection (T) comme quatrième métrique pour évaluer les performances de notre système. T représente le temps moyen nécessaire pour catégoriser un fichier PE de l'ensemble de test, et il est exprimé en secondes par fichier. Les résultats d'évaluation obtenus sont présentés dans les tableaux 4.7, 4.8, et 4.9.

APIs	ϕ	**nb. APIs**	**Classifieur**	**TD**	**TFP**	**PR**	**T**
A1	≥ 0.75	5	J48	100.00%	28.57%	88.99%	0.076
			B-J48	100.00%	28.57%	88.99%	0.076
			Ran-F	100.00%	28.57%	88.99%	0.076
			Rot-F	100.00%	28.57%	88.99%	0.076
			NB	100.00%	28.57%	88.99%	0.076
A2	≥ 0.5	11	J48	95.52%	11.90%	92.66%	0.078
			B-J48	97.01%	7.14%	95.41%	0.078
			Ran-F	98.51%	14.29%	93.58%	0.078
			Rot-F	**98.51%**	**7.14%**	**96.33%**	**0.081**
			NB	98.51%	28.57%	71.56%	0.078
A3	≥ 0.25	297	J48	94.03%	11.90%	91.74%	0.087
			B-J48	97.01%	9.52%	94.50%	0.087
			Ran-F	98.51%	14.29%	93.58%	0.086
			Rot-F	97.01%	9.52%	94.50%	0.0110
			NB	71.64%	9.52%	87.90%	0.086
A4	> 0 ($D^2 > 3.84$)	297	J48	97.01%	11.90%	93.58%	0.109
			B-J48	98.51%	9.52%	95.41%	0.114
			Ran-F	98.51%	14.29%	93.85%	0.109
			Rot-F	98.51%	19.05%	91.74%	0.172
			NB	62.69%	9.52%	73.39%	0.109
A5	-	1640	J48	95.52%	9.52%	93.58%	0.138
			B-J48	97.01%	9.52%	94.50%	0.147
			Ran-F	100.00%	11.90%	95.41%	0.136
			Rot-F	97.01%	9.52%	94.50%	0.343
			NB	64.18%	9.52%	74.31%	0.136

Tableau 4.7 – Résultats expérimentaux en utilisant les APIs uniquement.

Les résultats présentés dans le tableau 4.7 montrent que notre système est plus précis avec le sous-ensemble A2 et le classifieur Rot-F. En comparaison avec le sous-ensemble A5 on

constate une amélioration de 1,27% en matière de précision. Le temps de détection a été réduit de 40%, passant de 0.136s (A5 + Ran-F) à 0.081s (A2 + Rot-F).

IPEs	ϕ	**nb. IPEs**	**Classifieur**	**TD**	**TFP**	**PR**	**T**
H1	≥ 0.75	11	J48	100.00%	9.52%	96.33%	0.074
			B-J48	100.00%	9.52%	96.33%	0.076
			Ran-F	100.00%	9.52%	96.33%	0.074
			Rot-F	98.51%	7.14%	96.33%	0.074
			NB	100.00%	11.90%	95.41%	0.074
H2	≥ 0.5	14	J48	98.51%	9.52%	95.41%	0.074
			B-J48	97.01%	7.14%	95.41%	0.076
			Ran-F	100.00%	9.52%	96.33%	0.074
			Rot-F	**100.00%**	**7.14%**	**97.25%**	**0.077**
			NB	100.00%	11.90%	95.41%	0.074
H3	≥ 0.25	22	J48	98.51%	9.52%	95.41%	0.074
			B-J48	95.52%	9.52%	93.58%	0.074
			Ran-F	98.51%	7.14%	96.33%	0.074
			Rot-F	98.51%	9.52%	95.41%	0.076
			NB	100.00%	11.90%	95.41%	0.074
H4	> 0 ($D^2 > 3.84$)	50	J48	98.51%	9.52%	95.41%	0.077
			B-J48	95.52%	7.14%	94.50%	0.077
			Ran-F	97.01%	7.14%	95.41%	0.077
			Rot-F	98.51%	9.52%	95.41%	0.079
			NB	100.00%	16.67%	93.58%	0.0.077
H5	-	590	J48	98.51%	9.52%	95.41%	0.081
			B-J48	85.07%	4.76%	88.99%	0.086
			Ran-F	89.55%	4.76%	91.74%	0.081
			Rot-F	97.01%	7.14%	95.41%	0.116
			NB	100.00%	11.90%	95.41%	0.081

Tableau 4.8 – Résultats expérimentaux en utilisant les IPEs uniquement.

Nous pouvons voir dans les résultats présentés dans le tableau 4.8 que notre système a la meilleure précision (PR) avec le sous-ensemble H2 et le classifieur Rot-F(97,25%), avec une amélioration de 1,84% par rapport au degré de précision obtenu avec le sous-ensemble H5 (pas de sélection d'attributs). En outre, le temps moyen de détection a également été légèrement réduit (-0.004s).

En comparant les résultats dans le tableau 4.7 et le tableau 4.8, nous pouvons voir que les IPEs fournissent une meilleure précision (97,28%) par rapport aux APIs (96,20%), et un meilleur temps de détection (-0.004s). Dans le tableau 4.9, nous présentons les résultats de la combinaison des API et IPEs. Notez que seulement les résultats du meilleur classifieur ont été présentés.

Sous-ensembles	Classifieur	TD	TFP	PR	T
H1+A1	B-J48	100.00%	9.52%	96.33%	0.080
H1+A2	J48	100.00%	7.14%	97.25%	0.080
H1+A3	RAN-F	100.00%	9.52%	96.33%	0.089
H1+A4	B-J48	100.00%	7.14%	97.25%	0.143
H2+A1	B-J48	97.01%	7.14%	95.41%	0.080
H2+A2	B-J48	95.52%	7.14%	94.50%	0.080
H2+A3	**B-J48**	**100.00%**	**4.76%**	**98.17%**	**0.090**
H2+A4	J48	98.51%	7.14%	96.33%	0.144
H3+A1	ROT-F	100.00%	7.14%	97.25%	0.081
H3+A2	B-J48	95.52%	7.14%	94.50%	0.080
H3+A3	**B-J48**	**100.00%**	**4.76%**	**98.17%**	**0.092**
H3+A4	Ran-F	100.00%	7.14%	97.25%	0.144
H4+A1	J48	98.51%	9.52%	95.41%	0.081
H4+A2	NB	100.00%	9.52%	96.33%	0.082
H4+A3	Rot-F	98.58%	7.14%	96.33%	0.172
H4+A4	Rot-F	100.00%	7.14%	97.25%	0.232
H5+A5	B-J48	98.51%	4.76%	97.25%	0.187

Tableau 4.9 – Résultats expérimentaux en utilisant la combinaison des APIs-IPEs.

Nous pouvons voir à partir des résultats obtenus, que nous avons pu améliorer la précision de notre système, par rapport aux IPEs (+ 0,67%), et les APIs (+ 1,84%) et ce, en combinant les sous-ensembles H2 et A3 en utilisant le classifieur B-J48 (AdaBoostM1 + J48). Nous avons réussi à garder un bon temps de détection avec une moyenne de 0.090s. Nous avons également réussi à obtenir la même précision avec la combinaison des sous-ensembles H3 et A3 utilisant le même classifieur, mais avec un temps de détection de 0.092s. Par conséquent, nous préférons considérer la première combinaison (H2 + A3). D'après les résultats présentés précédemment, nous pouvons conclure que la méthode de sélection des attributs proposée a eu une contribution très importante dans l'augmentation de la précision du système et dans la réduction considérable du nombre d'attributs, et par conséquent, du temps de détection.

4.2.7 Comparaison des résultats et discussion

Dans cette section, nous allons évaluer l'efficacité de notre système en comparant nos résultats à ceux des systèmes précédemment cités, tel que présenté dans le tableau 4.10 et la figure 4.9.

Méthode	Attributs utilisés	TD	PR
Méthode proposée	APIs + IPEs	100%	98.17%
Schultz et al.[Schultz et al., 2001]	Chaines de caractères	97.43%	97.11%
Wang et al. [Wang et al., 2009b]	APIs	94.4%	93.71%
PE-Miner [Shafiq et al., 2009]	Informations structurelles	99%	99.1%
CIMDS [Ye et al., 2010]	APIs	88.16%	67.5%
Ding et al. [Ding et al., 2013]	APIs	97.3%	91.2%
Salehi et al. [Salehi et al., 2014]	APIs + Arguments	99.2%	98.4%

Tableau 4.10 – Comparaison de nos résultats avec ceux obtenus par d'autres méthodes.

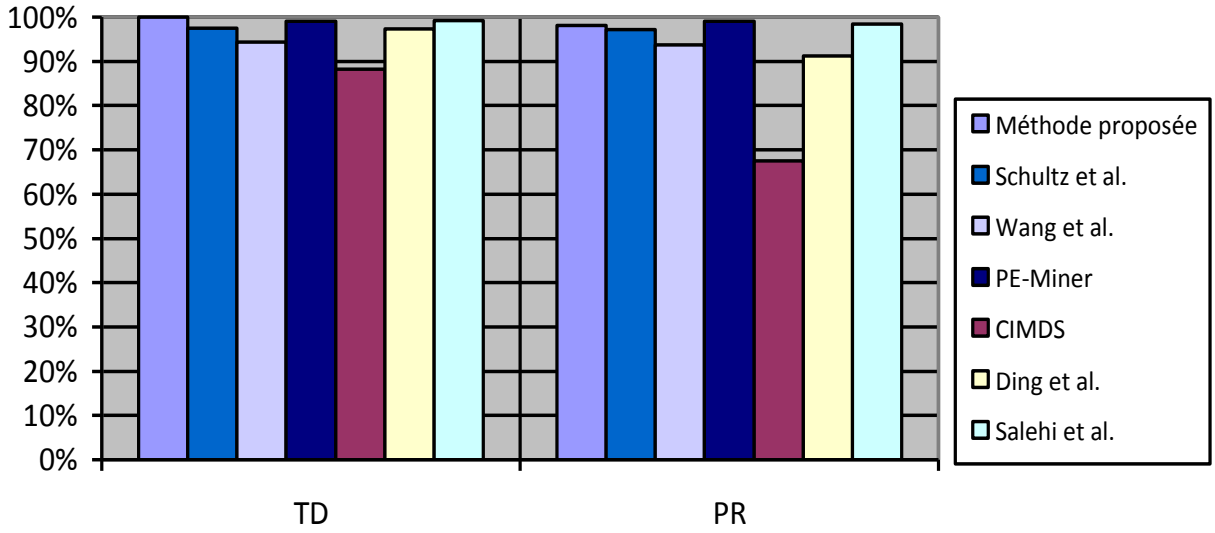

FIGURE 4.9 – Comparaison de nos résultats avec ceux obtenus par d'autres méthodes.

D'après les résultats présentés dans le tableau ci-dessus, nous pouvons voir que notre système surpasse tous les autres en matière de taux de détection (TD), vu qu'il a pu détecter 100% des malwares. En matière de précision (PR), notre système a le troisième meilleur taux derrière PE-Miner [Shafiq et al., 2009] (+ 0.93%) et le système proposé par Salehi et al.[Salehi et al., 2014] (+0.23%) ; cependant, il surpasse les quatre autres systèmes avec une amélioration qui varie de 1,06% à 30,67%.

En ce qui concerne le temps de détection (T), nous pouvons conclure que notre système est bien adéquat pour la détection en temps réel des logiciels malveillants, car il nécessite seulement 0.090s en moyenne pour l'ensemble du processus de catégorisation. Notre système vient ainsi égaler le temps de détection de CIMDS [Ye et al., 2008], et il est presque trois fois plus rapide que PE-Miner [Shafiq et al., 2009], puisque celui-ci nécessite 0.244s pour catégoriser un fichier.

La méthode proposée pour l'extraction et le pré-traitement des attributs nécessite respectivement 0.040s, 0.037s et 0.041s pour les API, les IPEs, et les API + IPEs. Ces résultats sont très satisfaisants par rapport à la méthode utilisée dans [Salehi et al., 2014], qui doit exécuter le programme analysé pendant 2 minutes afin d'en extraire les appels d'APIs et leurs arguments, ce qui représente près de 3000 fois le temps nécessaire pour notre méthode.

Le temps nécessaire pour générer le fichier .arff représente plus de la moitié du temps nécessaire pour la catégorisation. Cela dépend étroitement du nombre d'attributs utilisés. Par exemple, il a fallu 0.037s avec A1 + H1, 0.049s avec A3+ H3, et 0.142s pour A5 +H5. Notez que le processus de génération des fichiers .arff est lié à l'utilisation de l'environnement WEKA. Par conséquent, le temps nécessaire à cette opération ne sera pas pris en considération lorsque le classificateur sera directement implémenté et intégré dans notre système, ce qui permettra de réduire considérablement le temps de catégorisation.

Le temps nécessaire pour le processus de classification dépend du nombre d'attributs utilisés, ainsi que de l'algorithme de classification. Par exemple, avec les sous-ensembles H1+A1 et le classifieur B-J48, le processus de classification a pris 0.001s, et 0.004s avec H2 + A3 en utilisant le même classifieur. Il a fallu 0.119s et avec le même sous-ensemble et le classifieur Rot-F.

4.2.8 Épilogue

Dans cette première section, nous avons présenté un système de détection des malwares PE, qui est basé sur l'analyse des appels d'APIs et les IPEs. Le test du Khi^2 a été utilisé comme méthode de sélection des attributs, combiné avec le coefficient ϕ, qui a été utilisé pour sélectionner le nombre optimal d'attributs. Différents algorithmes de classification ont été utilisés pour évaluer notre système. Les résultats montrent ce dernier offre une meilleure précision (98.27%) en combinant les APIs et les IPEs en utilisant le classifieur J48 avec l'algorithme AdaBoostM1. Notre système est automatique et peut être utilisé pour la détection en temps réel des logiciels malveillants.

Dans le futur, et se restreignant à la méthode de sélection d'attributs à base de Khi^2, nous allons essayer d'augmenter la précision de notre système en combinant d'autres types d'attributs tels que les codes opérations, tout en essayant de garder la même rapidité. Dans une deuxième partie, nous projetons d'inclure un module pour la décompression des programmes. Cependant, nous nous somme engagé pour proposer une nouvelle approche permettant la catégorisation multi-classes de malwares par extraction d'associations d'APIs. Ceci sera présenté dans la section juste après.

4.3 Vers une catégorisation des malwares par association d'importations basée sur l'analyse des correspondances multiples (ACM).

4.3.1 Prologue

Dans la section précédente, nous avons présenté une méthode pour la détection binaire des malwares (malware ou bénin). Cependant, si on veut effectuer une catégorisation multi-classes, c'est-à-dire, déterminer à quelle catégorie appartient un malware (Virus, Ver, Cheval de Troie, etc.), l'utilisation d'une méthode bi-variée (telle la méthode du Khi^2) serait assez lourde et inadéquate pour la détection en temps réel, et cela en raison du grand nombre de types de malwares existants. De ce fait, nous introduisons une nouvelle méthode pour l'extraction des associations d'APIs, pour la catégorisation multi-classes des logiciels malveillants. Notre méthode vise à identifier les différentes APIs qui sont susceptibles d'être utilisées par les différentes catégories de logiciels malveillants. Pour ce faire, nous avons utilisé une méthode d'analyse de données multi-variée, à savoir, l'analyse des correspondances multiples (ACM). La méthode de l'ACM est généralement utilisée en sciences sociales [Le Roux and Rouanet, 2010], et à notre connaissance, n'a pas été précédemment utilisé dans la détection des malwares PE.

4.3.2 Méthode proposée

La méthode proposée pour l'extraction des associations d'APIs est basée sur l'observation qu'il existe des APIs qui sont plus susceptibles d'être utilisées en groupe par des logiciels malveillants et d'autres qui sont plus susceptibles d'être utilisées en groupe par les programmes bénins [Belaoued and Mazouzi, 2015a, Belaoued and Mazouzi, 2015c]. En outre, nous supposons que ces APIs peuvent varier d'un type de malware à un autre. Par conséquent, il serait très utile de pouvoir identifier automatiquement ces différentes associations et cela dans une unique analyse. À cette fin, nous présentons notre méthode, qui est basée sur l'ACM pour identifier ces associations d'APIs, qui sont spécifiques à certaines catégories de logiciels malveillants.

L'ACM vise à mettre en évidence les relations (de dépendance ou d'indépendance) entre des variables catégorielles. Cette méthode est basée sur le concept de questionnaire, qui est une matrice d'individus $\times$ Questions (variables catégorielles). Dans la terminologie de l'ACM, l'ensemble des PEs utilisés représentent les individus. Les variables catégorielles (qui représente dans notre cas les APIs) qui décrivent ces individus sont les questions, et les

réponses à ces questions sont appelées modalités. Le questionnaire est ensuite représenté sous forme d'un tableau, qu'on appelle tableau brut de données. Pour un ensemble de données, l'ACM projette chaque individu et réponse dans un système de coordonnées cartésien selon des axes principaux triés par ordre d'importance (contribution à l'explication de l'information). Ces axes condensent l'information contenue dans le jeu de données de sorte que la majorité de l'information est expliquée par seulement quelques-uns d'entre eux. Les axes feront également ressortir les variables qui décrivent le mieux l'ensemble des individus. Ainsi, les individus et les questions connexes sont tracés près les uns des autres dans l'espace des axes principaux. Habituellement, l'interprétation de l'ACM commence par l'analyse des projections obtenues afin de décrire les principaux axes en déterminant leurs significations. La prochaine étape consiste à identifier quelles sont les variables qui contribuent le plus à l'inertie des axes et identifier les groupes de variables corrélées[Le Roux and Rouanet, 2010].

Notre méthode est menée en trois phases, qui sont l'extraction de données, la représentation des données, et l'interprétation des résultats. La première phase, consiste à extraire les appels d'APIs à partir des fichiers PE. La deuxième phase, consiste à générer le tableau brut des données de l'ACM. Enfin, la dernière phase, consiste à analyser les résultats obtenus et d'identifier les différentes associations d'APIs. Afin d'effectuer l'analyse ACM, nous avons utilisé le logiciel Statistica [11], qui est un outil pour l'analyse de données statistiques .

Pour rester fidèle aux orgines de l'ACM, qui se trouvent encrées dans les sciences humaines, nous allons essayer d'expliquer son principe par un exemple qui a été présenté par Chakradeo et al. [Chakradeo et al., 2013]. Cet exemple illustre une enquête effectuée sur un ensemble de sept restaurants (R1,R2,...,R7) dans le but de déterminer le standing de ces derniers en se basant sur certains critères tels que le prix, la tranche d'âge des clients qui les fréquentent, etc. Le tableau 4.11 présente les résultats du questionnaire remis aux sept restaurants hypothétiques. Le questionnaire comporte six questions (qui représentent les variables de l'ACM), qui sont le tarif (cost) du restaurant, qui possède trois modalités qui sont : élevé (High) et moyen (Med), et bas (Low). La deuxième question représente la nature du parking et qui possède deux modalités : avec voiturier (Valet) et stationnement standard (Lot). La troisième question définit le style vestimentaire des clients (Attire) et qui possède deux modalités : formel (Formal) et décontracté (Causual). La quatrième question représente la tranche d'age de la clientèle ciblée (Ages) qui a aussi deux modalités : 18 ans et plus (+18) et toutes tranches d'ages confondues (All). Les deux dernières questions représentent respectivement la disponibilité du service de livraison (Delivery) et la télévision et qui ont deux modalités : OUI ou NON.

La figure 4.10 illustre la représentation graphique des réponses (caractéristiques de res-

11. Disponible à :`http://www.statsoft.com/Products/STATISTICA-Features`

Restaurant	Cost	Parking	Atire	Ages	Delivery	Television
R1	High	Valet	Formal	18+	NO	NO
R2	Med	Valet	Formal	All	NO	NO
R3	Med	Lot	Casual	18+	NO	YES
R4	Med	Valet	Casual	18+	NO	YES
R5	Med	Valet	Casual	All	NO	NO
R6	Low	Lot	Casual	All	YES	NO
R7	Low	Lot	Casual	All	NO	NO

Tableau 4.11 – Ensemble de données de l'ACM pour l'exemple des restaurants.

taurants) sur les deux premiers axes principaux des individus (restaurants) résultant de l'analyse de l'ACM. Le côté gauche de la figure 4.10 montre que les restaurants à tarifs bas (Lowcost) proposent souvent la livraison (Delvery) et sont familiaux. La présence de livraison vers l'extérieur du graphe, indique que c'est une caractéristique inhabituelle d'un restaurant, tandis que, les restaurants avec une tenue décontractée (Casual) ont tendance à être plus fréquents. De même, les restaurants à tarifs élevé (High Cost) sont moins fréquents, et sont fortement corrélés avec les vêtements formels (Formal) dans cet échantillon (qui se trouve dans le coin supérieur droit de la figure 4.10.

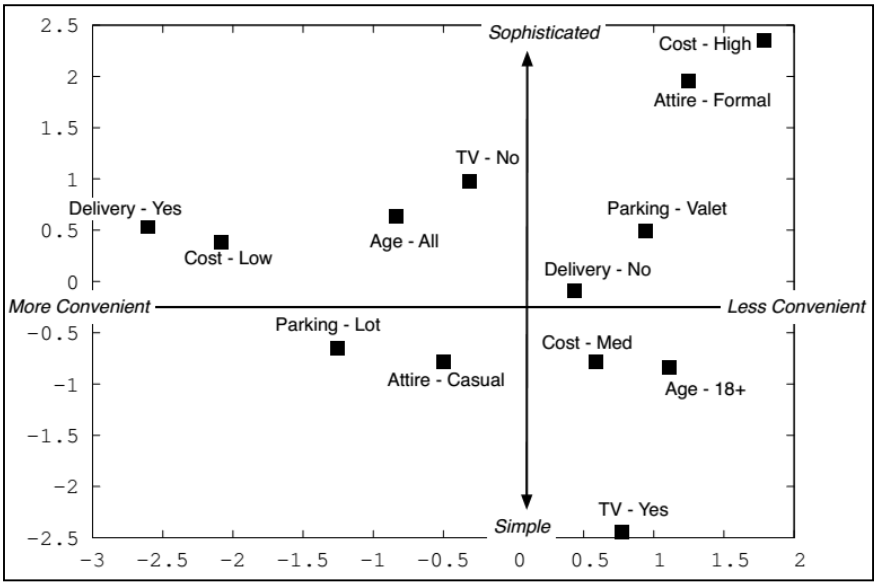

FIGURE 4.10 – Résultat de l'ACM pour l'exemple des restaurants(Corrélation entre les variables) [Chakradeo et al., 2013].

Dans une analyse basée sur l'ACM, l'interprétation des axes principaux permet de mieux décrire l'ensemble des réponses des individus. Dans cet exemple, les auteurs ont donné l'interprétation suivante pour les axes :

- L'axe horizontal reflète la commodité. En effet, les restaurants avec des caractéristiques situées à gauche de l'axe vertical ont tendance à être moins chers et plus accessibles, tandis que ceux à la droite de l'axe ont tendance à exiger plus d'efforts pour en profiter.
- L'axe vertical du graphique reflète l'ambiance ; les restaurants et les caractéristiques au-dessous de l'axe horizontal ont tendance à être plus simple, tandis que ceux au-dessus de l'axe horizontal ont tendance à être plus sophistiqué.

Dans notre cas, les caractéristiques des restaurants vont être remplacées par les types de malwares, ainsi que les APIs. De ce fait, nous allons essayer de regrouper les APIs et les types de malwares de la même façon que pour les caractéristiques des restaurants. En outre, l'interprétation des axes pourrait nous permettre de ressortir un caractère général pour les APIs. par exemple, celles qui sont utilisées en groupe par les malwares et celles utilisée en groupe par les fichiers bénins, ou par exemple les APIs qui sont plus utilisées par les malwares se propageant via un réseau , etc.

4.3.3 Expérimentation

Échantillons

Nous avons utilisé un échantillon de programmes PE qui est composé de 210 fichiers (120 malwares et 90 programmes bénins). Comme pour notre première approche présentée dans la section précédente, les programmes bénins sont composés de logiciels utilitaires et certains fichiers systèmes obtenus à partir d'une installation récente de Windows XP. L'ensemble des fichiers PE infectés ont été obtenus à partir de la collection VxHeavens [12] et sont composés de douze différents types de logiciels malveillants, comme indiqué dans le tableau 4.12.

Extraction et pré-traitement des APIs

Afin d'extraire les appels d'APIs, nous avons utilisé la même méthode d'extraction statique, basée sur l'analyse de l'IAT décrite précédemment. La phase de pré-traitement consiste également à enlever les APIs en double dans le même fichier PE, les regrouper, et les trier dans l'ordre descendant en fonction de leurs fréquences d'appel, et ce en considérant les fréquences dans les logiciels malveillants. Enfin, nous avons obtenu la liste d'API dont un apperçu est présenté dans le tableau 4.13.
A cause de certaines contraintes liées aux outils d'analyse, notamment à Statistica, nous nous sommes limités à analyser seulement les APIs qui ont été importés par au moins 30% des

12. Disponible à : `http://vxheaven.org/vl.php`

Type de malware	Nombre
Backdoor	10
Email-Worm	10
Exploit	10
Hacktool	10
Network-Worm	10
P2P-Worm	10
Trojan	10
Trojan-Downloader	10
Trojan-Dropper	10
Trojan-Spy	10
Virus	10
Worm	10
Total	120

Tableau 4.12 – Echantillon de malwares utilisé.

Code	API	Fréq. malware (120)	Fréq. bénins (90)
1	GetProcAddress	115 (4%)	68 (76%)
2	CloseHandle	114 (95%)	64 (71%)
3	GetModuleFileNameA	113 (94%)	31 (34%)
4	GetCommandLineA	113 (94%)	26 (29%)
5	WriteFile	113 (94%)	52 (58%)
...	...	...	...
1121	SendArp	1 (1%)	0 (0%)

Tableau 4.13 – Aperçu des APIs extraites.

logiciels malveillants. Par conséquent, nous avons obtenu une liste composée de 83 APIs (A1 à A83).

Génération du tableau brut de données

Tel l'aperçu présenté dans le tableau 4.14, le tableau brut de données de notre analyse est composé de 84 colonnes qui représentent les variables utilisées (PE + 83 APIs), et 210 lignes qui représentent les individus. Les variables qui représentent les appels d'APIs ont deux modalités 'Y' et 'N'. 'Y' signifie que l'API correspondante a été importée par le fichier PE correspondant (l'individu), et 'N' signifie que l'API n'a pas été importée. La variable PE possède treize modalités qui correspondent aux douze types de logiciels malveillants, plus la modalité 'bénin'.
Le tableau ci-dessus a été généré automatiquement en utilisant un script Python. Ce dernier

No.	PE	A1	A2	A3	A4	A5	...	A83
1	Backdoor	Y	Y	Y	Y	Y	...	N
2	Backdoor	N	Y	Y	Y	Y	...	N
...	...	...	...	...	...	...	...	...
11	E-Worm	Y	Y	Y	Y	Y	...	Y
12	E-Worm	Y	Y	Y	Y	Y	...	N
...	...	...	...	...	...	...	...	...
210	Bénin	Y	Y	N	N	N	...	N

Tableau 4.14 – Aperçu du tableau brut de données de l'ACM.

recherche les APIs précédemment sélectionnées (83 APIs) dans chaque fichier PE (malware et bénin) de l'échantillon. Le tableau est représenté au format CSV ("Comma-separated values", pour "valeurs séparées par des virgules") et sera donné en entrée au logiciel Statistica. Les résultats d'analyse obtenus sont présentés dans ce qui suit de la section.

Interprétation des résultats

Comme mentionné précédemment, la méthode de l'ACM vise à réduire et à condenser les informations selon un nombre réduit d'axes factoriels [Le Roux and Rouanet, 2010]. Par conséquent, la première étape à entreprendre quand on commence l'interprétation des résultats est de déterminer combien d'axes reflètent la majorité de l'information. Cela se fait généralement par l'analyse de la contribution à l'inertie des différents axes. Le tableau 4.15 montre les différentes valeurs propres (Eigen values) de l'analyse.

No.	Valeurs propres	pourc.Inértie	pourc.Cumul
1	0.31	30.89%	30.89%
2	0.18	17.54%	48.43%
3	0.06	5.95%	54.38%
4	0.05	4.78%	59.16%
5	0.04	4.11%	63.28%
...	...	...	...
83	0.00	0.00	100%

Tableau 4.15 – Aperçu des valeurs propres et l'inertie de l'analyse.

Comme nous pouvons le voir, les deux premières dimensions représentent 48,43% de l'inertie totale, qui est considérée comme une bonne valeur pour l'interprétation des résultats. On peut aussi constater qu'il y a un saut significatif dans le pourcentage d'inertie étant donné que l'axe 3, représente moins de 6% de l'information totale. Par conséquent, nous limiterons

notre analyse aux deux premiers axes qui seront représentés graphiquement par l'axe 1 (axe horizontal), et l'axe 2 (axe vertical). Après avoir sélectionné les axes qui seront utilisés pour l'interprétation de nos résultats, nous devons identifier ce qu'ils représentent. À cette fin, nous devons analyser la projection obtenue, qui est présentée dans figure 4.11. Notez que dans ce travail, nous allons présenter une interprétation purement graphique des résultats, ce qui sera fait en deux phases. Tout d'abord, nous devons identifier ce qui représente les deux axes. La deuxième phase de l'analyse consistera à identifier les groupes d'API associés, en les regroupant avec le type de malwares le plus proche.

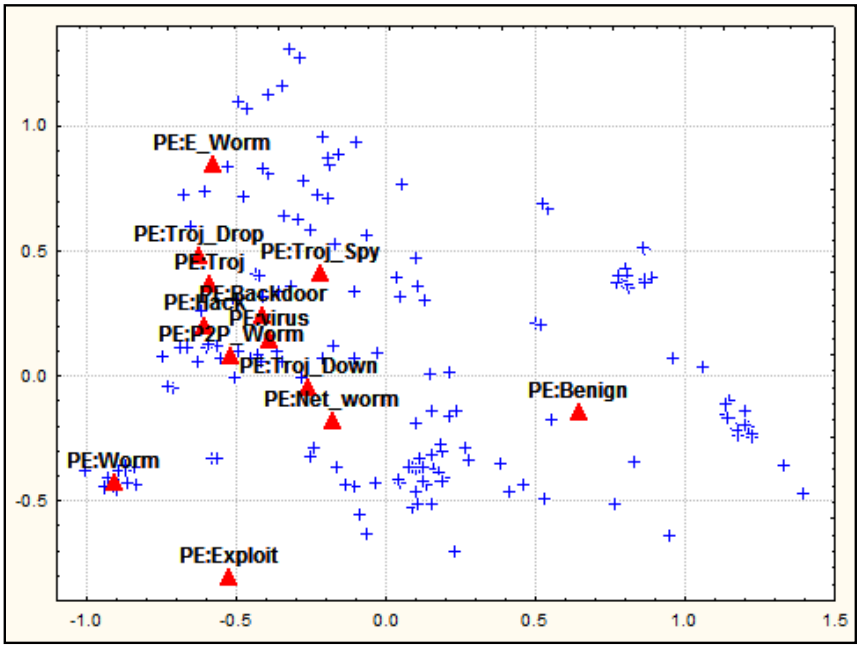

FIGURE 4.11 – Représentation graphique des résultats d'analyse de l'ACM.

La figure 4.11, représente la projection des différentes APIs (croix bleues) et les différentes catégories de logiciels malveillants (triangles rouges). Nous pouvons clairement voir, que la modalité *'bénin'* est tracée du côté positif de l'axe 1 et toutes les autres modalités qui représentent les catégories de programmes malveillants sont tracées du côté négatif de l'axe. Par conséquent, nous pouvons conclure que l'axe 1 sépare les deux catégories de fichiers PE (i.e., malwares et bénins). Le deuxième axe (axe vertical) peut également être utilisé pour séparer entre les APIs qui sont susceptibles d'être utilisées par des logiciels malveillants et les APIs qui sont susceptibles d'être utilisées par les programmes PE bénins. En effet, huit des douze types de logiciels malveillants sont tracés du côté positif de l'axe, et seulement quatre d'entre eux sont tracées dans le coté négatif, où la modalité *'bénin'* est également tracée.

Selon notre interprétation des axes, ainsi que le principe de l'ACM, les APIs qui ont la plus forte contribution à l'inertie de l'axe 1 et 2 devraient avoir la plus haute importance dans

le processus de détection des logiciels malveillants (classification binaire malware/bénin). La force de la contribution de chaque variable (c-à-d chaque API) à l'inertie d'un axe est proportionnelle à la valeur de la cos^2 de l'angle entre l'axe et de la modalité (présence ou pas de l'API). Cette dernière valeur est comprise entre 0 et 1. Dans le tableau 4.16, nous allons présenter les 10 APIs avec les valeurs les plus élevées du cos^2 pour les deux axes, donc celles ayant la plus grande importance pour la détection binaire des malwares.

Axe 1		**Axe 2**	
API	Cos^2	**API**	Cos^2
Ggetcpinfo	0.85	Raiseexception	0.49
Virtualalloc	0.82	Createwindowexa	0.48
Virtualfree	0.80	Leavecriticalsection	0.47
Getstringtypew	0.76	Entercriticalsection	0.47
Sethandlecount	0.75	Tlssetvalue%	0.47
Lcmapstringa	0.75	Defwindowproca	0.47
Getenvironmentstrings	0.75	Tlsgetvalue	0.46
Getstringtypea	0.74	Initializecriticalsection	0.45
Freeenvironmentstringsw	0.74	Deletecriticalsection	0.45
Getenvironmentstringsw	0.74	Dispatchmessagea	0.42

Tableau 4.16 – Les 10 APIs ayant les plus importantes contributions à l'inertie des deux axes.

La deuxième phase de l'analyse, consiste à identifier les différentes associations d'APIs qui sont spécifiques aux différentes catégories des codes malveillants. Cela, peut être fait en regroupant les APIs avec la catégorie de malware la plus proche. Pour ce faire, nous allons utiliser la métrique de Manhattan, qui est utilisée pour mesurer la distance entre un point 'a' qui représente dans notre cas une API donnée et un autre point 'b' qui représente un type de logiciels malveillants (virus, ver, cheval de Troie, etc.). La distance de Manhattan (d) est calculée en utilisant la formule suivante :

$$d(a,b) = \sum_{i=1}^{2} |a_i - b_i| \tag{4.9}$$

Après avoir calculé la distance d, nous allons regrouper les différentes APIs avec leur catégorie malware correspondante. Après l'attribution des API à leur type de malware correspondant, nous allons calculer la valeur moyenne (M) des distances pour les APIs correspondant à un type spécifique de malwares, et ne garder que les APIs qui ont une valeur d inférieure à M. Notre objectif est de retirer les APIs non pertinentes en ne gardant que les plus proches de leur catégorie de malwares. Le tableau 4.17 résume les associations d'APIs obtenues regroupées par types de logiciels malveillants.

Notez que les résultats obtenus ne comprennent pas les malwares de type 'Exploits', puisque notre analyse n'a pas été en mesure d'identifier les associations d'APIs pour cette catégorie de logiciels malveillants, comme le montre la figure 4.11. Ceci peut être expliqué par le fait que ce type de malwares ont un comportement assez proche (similaire) d'un autre type de malwares tels que les Chevaux de Troie, par exemple.

4.3.4 Épilogue

Dans ce travail, nous avons présenté une nouvelle méthode pour l'extraction des associations d'APIs pour la catégorisation multi-classes des malwares PE. Notre méthode, qui est basée sur la méthode de l'ACM, a montré de très bons résultats, puisque nous étions en mesure d'identifier les différentes associations d'APIs pour les 11 types de programmes malveillants (des 12 types initialement utilisés). Comme travaux futurs, nous allons essayer d'automatiser la méthode proposée par sa mise en œuvre à l'aide d'un environnement de programmation pour l'analyse des données tel que *R*. Dans ce travail, les résultats obtenus ont été analysés et nos conclusions ont été établies selon les principes de l'ACM. Cependant, notre méthode doit être validée par son déploiement dans un système de détection de logiciels malveillants, afin d'évaluer la précision des résultats obtenus.

4.4 Conclusion

Le long de ce chapitre, nous avons présenté deux approches différentes pour la catégorisation des codes PE, basées sur des méthodes statistiques décisionnelles. La première, utilise le test d'hypothèse KHI^2, en exploitant différentes informations se trouvant dans les entêtes, notamment les importations des API Windows. La seconde, utilise l'analyse factorielle des correspondances multiples, pour extraire les associations d'APIs, caractérisant un code PE malveillant. Dans les deux cas, les méthodes proposées se sont avérées efficaces, et notamment temps-réel. Ces méthodes de catégorisation, nous seront utiles pour la proposition d'une approche collective et consensuelles pour la détection de malwares dans les réseaux étendus, objet du prochain chapitre.

Malware	**APIs**	
Backdoor	CreateFileA RegSetValueexA	GetVersionExA Getlocaltime
Trojans	DeleteFileA	
Troj-Drop	Setendoffile Getsysteminfo	Getlocaleinfoa
Troj-Down	Getlasterror Unhandledexceptionfilter	Getmodulehandlea Getcurrentprocess
Trojan-Spy	Sleep Readfile Regclosekey Createthread Getcurrentthreadid	Waitforsingleobject Interlockedincrement nterlockeddecremen Regcreatekeyexa Regqueryvalueexa
Worm	Freeenvironmentstringsa Freeenvironmentstringsw Getenvironmentstringsw Getenvironmentstrings Getstringtypew Getoemcp Lcmapstringa	Setstdhandle Heapfree Sethandlecount Getstringtypea lcmapstringw Heapdestroy Flushfilebuffers
E-Worm	Messageboxa Comparestringa Leavecriticalsection Entercriticalsection Getfilesize Initializecriticalsection Findfirstfilea	Findclose Dispatchmessagea Deletecriticalsection Virtualquery tlsgetvalue tlssetvalue
Virus	Getprocaddress Closehandle Writefile	Loadlibrarya Getstartupinfoa Multibytetowidechar
Hacktool	Getcommandlinea Rtlunwind Virtualalloc	Virtualfree Getfiletype Multibytetowidechar
P2P Worm	Getmodulefilenamea Getstdhandle Widechartomultibyte Getcpinfo	Exitprocess Getfiletype Translatemessage Copyfilea
P2P Worm	Terminateprocess	

Tableau 4.17 – Résultats de l'analyse après le regroupement des APIs à l'aide de la distance de Manhattan.

Chapitre 5

Vers une Approche Collective et Collaborative pour la Détection des Malwares

Sommaire

5.1 Introduction

Dans ce dernier chapitre, nous présentons notre deuxième contribution qui consiste en la proposition d'une approche utilisant les systèmes multi-agents (SMA), pour la détection collective et coopérative des malwares PE. Notre but est d'élever le degré de certitude d'une analyse en faisant collaborer des outils de détection de malwares de natures hétérogènes, installés

sur différents nœuds d'un réseau afin d'arriver à une décision collaborative et consensuelle à propos d'un code PE donné (s'il s'agit d'un malware ou non). Notre approche s'inspire de l'architecture des systèmes de détection d'intrusion collaboratifs (CIDS, pour Collaborative Intrusion Detection System) distribués (qui sera présentée dans la section suivante) pour la prise de décision. En effet, nous avons utilisé les systèmes multi-agents afin d'intégrer le processus de collaboration entre les nœuds du réseau. Ceci va nous permettre de remédier aux problèmes liés à la partialité de l'information échangée, ainsi que le manque d'interaction entre les nœuds, dont souffrent les actuels CIDS [Zhou et al., 2010, Nafir et al., 2014].

Selon notre approche, chaque nœud possède son propre agent qui est situé, possédant un ensemble de rôles et réagissant à un ensemble d'événements. Ce dernier va se charger d'analyser le fichier à l'aide de l'outil de détection de malwares installé au niveau de son nœud, et établir une décision, dite locale, qui sera ensuite combinée avec les résultats de décision, obtenus à partir des autres nœuds du réseau afin d'aboutir à une décision globale et consensuelle. Dans ce qui suit, nous allons présenter l'approche en question, mais avant cela nous allons d'abord introduire les différents types de CIDS existants, en présentant les avantages et inconvenants de chaque type, ainsi qu'un aperçu sur les systèmes multi-agents.

5.2 Les systèmes de détection d'intrusion collaboratifs (CIDS)

Les systèmes de détection d'intrusion collaboratifs (CIDS), ont été proposés afin de faire face aux limites des IDS classiques, liées à la détection des attaques coordonnées, telle que l'attaque DDoS [Zhou et al., 2010, Vasilomanolakis et al., 2015]. Contrairement aux IDSs classiques qui agissent de manière isolée, les CIDS ont le potentiel de détecter les intrusions qui se produisent dans l'ensemble du réseau Internet en même temps. Cela est fait en combinant les résultats d'analyses provenant de différents IDS implantés au niveau des nœuds ou des sous-réseaux. Un CIDS se compose de deux unités fonctionnelles, principales qui sont l'unité de détection et l'unité de corrélation (d'analyse) [Zhou et al., 2010, Bye et al., 2010, Vasilomanolakis et al., 2015].

- **L'unité de détection :** Elle est composée d'un ou plusieurs capteurs qui ont pour but de surveiller l'activité au niveau d'un sous-réseau ou d'une machine, et génère des alertes d'intrusion de bas niveau.
- **L'unité de corrélation :** Elle se charge de la détection effective des intrusions, et cela en procédant à l'analyse des alertes d'intrusions obtenues à partir de l'unité d'analyse.

Il existe plusieurs critères pour la classification des CIDS ; cependant, nous allons considérer la classification proposée par C. V. Zhou et al. [Zhou et al., 2010] ainsi que E.Vasilomanolakis et al. [Vasilomanolakis et al., 2015] et qui divisent les CIDS selon leur architecture en trois catégories, qui sont les CIDS centralisés, CIDS hiérarchiques, et les CIDS distribués. Selon cette décomposition, un CIDS est généralement composé d'une ou plusieurs unités de détection, ainsi qu'une ou plusieurs unités de corrélation.

5.2.1 Les CIDS centralisés

Les CIDS centralisés, utilisent un ensemble d'unités de détection et une seule unité de corrélation (nœud central) sur laquelle va converger tout le flux d'information généré à partir des unités de détection (figure 5.1). Ce nœud central a pour rôle la collecte d'informations ainsi que l'analyse, afin de déterminer s'il s'agit d'une menace ou pas. Comme toute architecture centralisée, cette approche est facile à mettre en œuvre et adéquate pour un déploiement à petite échelle, telle que dans les petites entreprises. Ceci dit, elle souffre d'un inconvénient majeur qui réside dans le nœud central. En effet, le processus de corrélation s'arrêtera avec la mise hors service du nœud central. En plus, la capacité de traitement du nœud central va certainement limiter le volume de données pouvant y être traité. Cependant, les attaquants habiles peuvent par exemple lors d'une attaque de type DDoS esquiver ce genre de systèmes, et ce, en réduisant l'intensité du trafic lors de l'attaque d'un réseau donné.

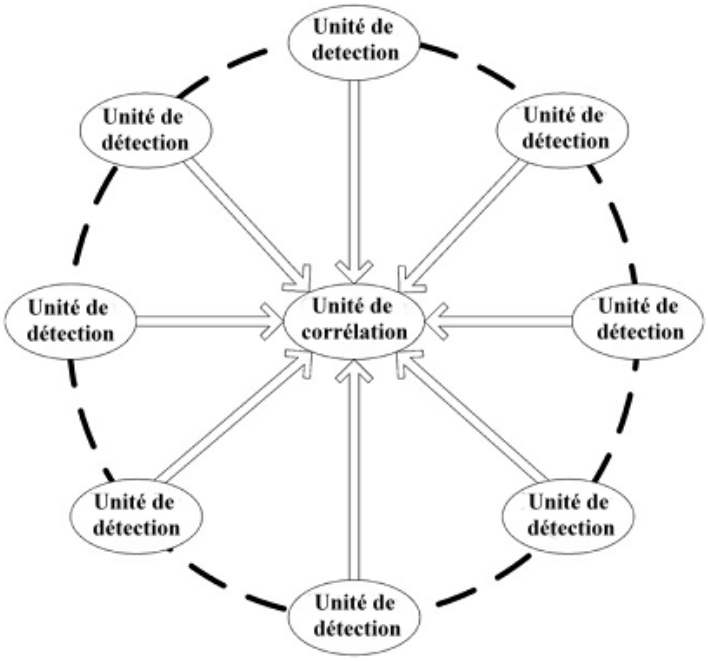

FIGURE 5.1 – Architecture des CIDS centralisés [Zhou et al., 2010]

5.2.2 Les CIDS hiérarchiques

Les CIDS hiérarchiques, aussi appelés CIDS décentralisés, ont été introduits afin de remédier au problème du nœud central dans les architectures centralisées. Ce type d'architecture repose sur le principe de hiérarchisation du réseau suivant un critère donné tels que la répartition géographique ou administrative, l'homogénéité des plateformes logicielles, etc [Zhou et al., 2010]. Chaque sous-groupe de la hiérarchie possède son propre nœud central au niveau duquel va s'effectuer l'analyse. Les informations générées à partir de ce nœud, vont être transmises à un nœud supérieur hiérarchiquement (figure 5.2). Cette approche vient réduire la dépendance de la totalité du réseau au nœud central dans une architecture centralisée. Cependant, les nœuds centraux qui se situent à un niveau hiérarchique élevé, restent les maillons faibles de ce type de systèmes.

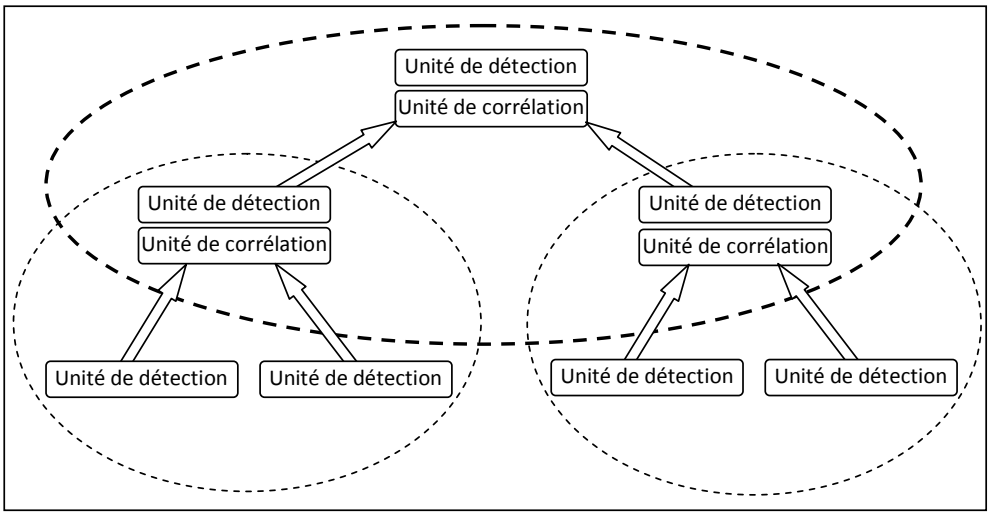

FIGURE 5.2 – Architecture des CIDS hiérarchiques [Zhou et al., 2010]

5.2.3 Les CIDS distribués

Ce genre d'architecture ne comporte aucune entité centrale, donc elle élimine complètement les inconvénients des approches centralisées et hiérarchiques (figure 5.3). Dans une architecture distribuée, chaque nœud procède à l'analyse de la menace à son niveau. Il échange ensuite les résultats de son analyse avec les nœuds de son voisinage. Ces architectures sont inspirés des réseaux pair à pair (P2P), donc, elles sont envisageables pour des réseaux large échelles tel qu'Internet.

Les CIDS totalement distribués, ont permis de résoudre le problème lié au nœud central. Cependant, ces derniers, souffrent d'un inconvénient qui réside dans leur faible taux

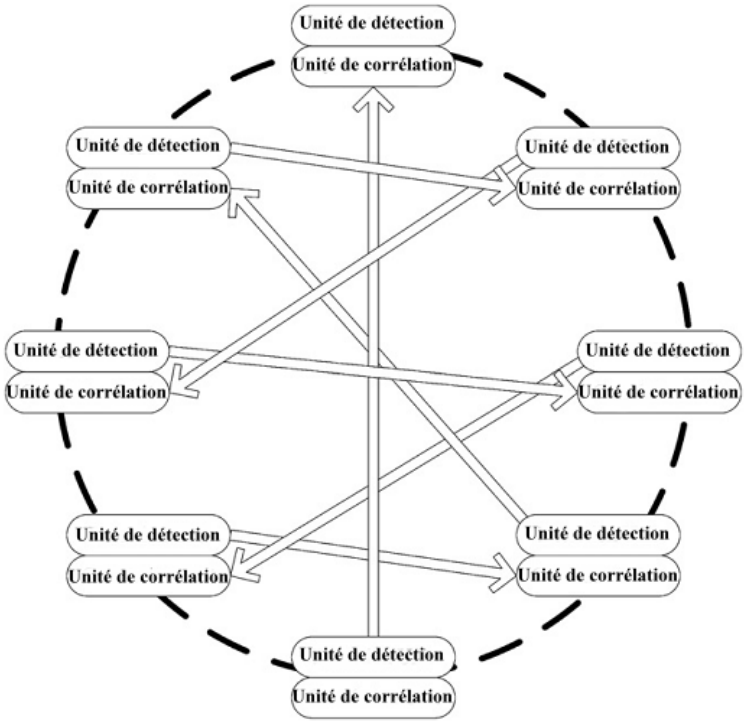

FIGURE 5.3 – Architecture des CIDS distribués [Zhou et al., 2010]

de précision, comparés aux CIDS centralisés [Zhou et al., 2010]. En effet, dans les CIDS distribués, ce n'est pas toutes les informations d'alertes qui peuvent-être accessibles (problème de partialité d'information) à l'endroit où la décision de détection est prise pour une attaque donnée. Afin de remédier à cet inconvénient, nous proposons un mécanisme de décision consensuelle. En effet, dans l'approche proposée, la décision finale émerge de l'ensemble des décisions locales. Notre choix des systèmes multi-agents, n'est donc pas fortuit du fait que ces derniers définissent le mieux les aspects qui nous intéressent tels que la collaboration, le traitement coopératif, et l'émergence de la décision [Ferber, 1995].

5.3 Les systèmes multi-agents (SMA)

5.3.1 Présentation et définitions

Depuis quelques années, les systèmes multi-agents (SMA) ont pris une place de plus en plus importante parmi la panoplie des paradigmes informatiques. Ces derniers ont diverses applications, telles que l'étude de phénomènes sociaux (ex. phénomènes de groupes, et coalition), l'ingénierie, les réseaux, et les systèmes distribués, etc. Les SMAs se situent à l'intersection de plusieurs domaines, en particulier ceux de l'intelligence artificielle, l'algorithmique distribuée,

et le génie logiciel [Drogoul, 1993, Chaib-Draa et al., 2001, Simonin and Ferber, 2003].

L'intelligence artificielle (I.A) étant un domaine qui modélise le comportement d'un seul agent, le passage du comportement individuel aux comportement collectif est devenu nécessaire pour combler les limites de l'I.A classique à résoudre des problèmes complexes. C'est ainsi que l'I.A Distribuée (I.A.D), branche de l'I.A est apparue [Boudaoud, 2002]. En effet, l'I.A.D a permis de distribuer la résolution de problèmes sur plusieurs entités (agents), en introduisant le concept de système Multi-Agents. Cette discipline s'intéresse aux comportements intelligents qui sont le produit de l'activité coopérative de plusieurs agents dont les principales caractéristiques sont : la communication, la coordination, et la coopération [Le Bars, 2003].

Plusieurs définitions ont été proposées pour la notion d'agent. J.Ferber [Ferber, 1995] définit un agent comme étant *Une entité autonome, réelle ou abstraite, capable d'agir sur elle-même et sur son environnement, et qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.* Quant à Wooldridge et al. [Wooldridge et al., 1995], ils définissent un agent comme étant *Un système informatique situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu.* Également selon J.Ferber [Ferber, 1995], on appelle agent une entité physique ou virtuelle :

- qui est capable d'agir dans un environnement,
- qui peut communiquer directement avec d'autres agents,
- qui possède des ressources propres,
- qui est capable de percevoir son environnement,
- qui possède des compétences et offre des services,
- qui peut éventuellement se reproduire,
- dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont il dispose, et en fonction de sa perception, de ses représentations et des communications qu'il reçoit.

5.3.2 Typologie des agents

Un agent peut appartenir à différentes catégories selon différents axes de classification. Cependant, nous allons considérer une classification des agents selon le processus de prise de décision, où un agent peut être soit cognitif, soit réactif, soit les deux en même temps (hybride) [Chaib-Draa et al., 2001, Guessoum, 2003, Le Bars, 2003, Mazouzi, 2008]. Dans ce qui suit une description de ces types d'agent.

- **Agents cognitifs (délibératifs) :** ils sont caractérisés par leur capacité de raison-

nement et de négociation ; ils utilisent des langages de haut niveau, leur permettant d'échanger leurs connaissances et coordonner explicitement leurs actions [Mazouzi, 2008].

- **Agents réactifs :** *'Il s'agit d'entités simples, qui réagissent d'une manière réactive aux stimulus qui proviennent de leur environnement, en changeant de comportement et/ou en agissant sur les objets de l'environnement'* [Mazouzi, 2008].
- **Agents hybrides :** les agents hybrides intègrent l'aspect cognitif et réactif, ils ont été proposés afin de d'améliorer les temps d'actions et de décision [Guessoum, 2003].

5.3.3 Interactions entre agents

'Une interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques' [Ferber, 1999]. L'interaction entre agents peut se résumer en la communication, la collaboration, la coordination, et la coopération [Ferber, 1995, Ferber, 1997].

La communication

La communication définit l'ensemble des processus physiques et psychologiques par lesquels s'effectue l'opération de mise en relation d'un agent émetteur avec un ou plusieurs agents récepteurs, dans l'intention d'atteindre les objectifs [Chaib-Draa et al., 2001]. Il existe deux méthodes de communication qui sont la communication par partage d'information, et la communication par envoie de message [Chaib-Draa et al., 2001].

- **Communication par partage d'informations (tableau noir) :** Il s'agit là, d'une communication indirecte où les agents peuvent communiquer à travers un espace de travail commun (partagé) appelé tableau noir (Blackboard en anglais), dont le mécanisme est de déposer l'information ou la connaissance dans cette zone, et ainsi l'agent concerné viendra récupérer cette information.
- **Communication par envoie de message :** dans ce type de communication les agents sont en liaison directe, les messages sont envoyés directement et explicitement aux destinataires. Il existe trois types de messages : les questions, les réponses et les informations. Au niveau protocolaire, un envoi de message peut être synchrone (un agent émetteur attend la réponse de son récepteur) et asynchrone (un agent émetteur peut agir immédiatement après avoir placé son message dans une fille d'attente). Les agents disposent de langages communs pour pouvoir coopérer pour la résolution d'un problème. Ces langages sont appelés langages de communication entre agents (ACLs, en anglais Agent Communication Languages). Il existe différents ACL, parmi eux nous citons : KQML (Knowledge Query and Manipulation Langage), KIF (Knowledge In-

terexchange Format) et ACL FIPA.

Collaboration

La collaboration s'intéresse à la manière de répartir le travail entre plusieurs agents, qu'il s'agisse de techniques centralisées ou distribuées[Ferber, 1997].

La coordination

La coordination analyse la manière dont les actions des différents agents doivent être organisées dans le temps et l'espace de manière à réaliser les objectifs [Ferber, 1997]. B. Chaib-draa et al. [Chaib-Draa et al., 2001] présentent la coordination entre agents par des exemples de la vie quotidienne tels que deux déménageurs déplaçant un meuble lourd, deux jongleurs échangeant des balles avec lesquelles ils jonglent, des personnes qui parlent à tour de rôle en se passant un micro, etc.

La Coopération

La coopération, est la forme la plus générale ainsi que la plus importante des interactions dans les systèmes multi-agents. En effet, la coopération est nécessaire pour la résolution distribuée d'un problème. Celle-ci se caractérise par l'activité d'un groupe d'agents convergeant vers un but global par l'achèvement de leurs propres buts locaux [Bond, 1990]. Cela peut donc se traduire par le fait de déterminer qui fait quoi, quand, où, avec quels moyens, de quelle manière et avec qui [Ferber, 1997]. Il existe plusieurs modèles de coopérations d'agents, parmi eux on trouve la coopération par partage de taches, de résultats intermédiaires, etc.

5.4 Travaux Similaires

Dans ce qui suit de cette section, nous survolons quelques travaux récents ayant proposé des approches collaboratives, pour la détection des malwares, et ce, afin que nous nous situons par rapport à ces travaux, et nous montrons par la suite, la particularité de notre travail, et la contribution qui nous y est propre.

Oberheide et al. [Oberheide et al., 2008] présentent une approche centralisée pour la détection collaborative des malwares. Cette approche est basée sur la technologie du Cloud Computing [1] (en français informatique nuagique). L'approche proposée permet ainsi d'analyser un fichier suspect à l'aide de plusieurs outils antivirus installés au niveau du Cloud. En

1. La technologie du Cloud Computing permet d'exploiter des ressources informatiques (puissance de calcul, espace de stockage, etc.) via des serveurs distants par l'intermédiaire du réseau internet.

effet, des agents locaux fonctionnant sur des appareils mobiles, agissent tels des programmes anti-virus en procédant au contrôle de l'activité des fichiers sur le système. Si un fichier identifié n'est pas disponible dans le cache des fichiers précédemment analysés, il sera envoyé au service réseau au niveau du Cloud. Le service réseau est responsable de la vérification des fichiers, et permet de déterminer si un fichier est malveillant ou non. L'utilisation en parallèle de moteurs antivirus multiples a permis d'améliorer grandement la précision de détection.

RAVE [Silva et al., 2010], est l'acronyme pour Replicated Antivirus for Email infrastructes, qui signifie *moteur antivirus répliqué pour infrastructures de courriel*. Ce dernier est un système centralisé pour la détection collaborative des malwares qui est déployé au niveau du réseau local se situant entre le réseau internet et l'infrastructure de courriel. Rave est composé d'un ensemble d'agents appelés *répliques*. Ces répliques comportent deux éléments qui sont le *Payload* et le *Wormhole*. Le premier se charge d'analyser les fichiers à l'aide d'un outil antivirus à la recherche de logiciels malveillants. Le second quant à lui, a pour but de collecter les résultats d'analyse du *Payload* et d'envoyer le résultat via courriel aux autres 'répliques'. Le système dispose aussi d'une entité centrale appelée *Wormehole maitre* qui a pour but la collecte des résultats de l'analyse des différents agents et la prise de décision. Cette dernière est basée sur un mécanisme de vote.

Ces deux premiers systèmes ont l'inconvénient de dépendre d'entités centrales pour la collecte des informations sur les menaces, ce qui les rend inadaptés pour le déploiement à large échelle.

ENDMal [Lu et al., 2013] est un système semi-centralisé (hiérarchique) collaboratif basé sur l'analyse dynamique pour la détection des malwares. ENDMAL dispose d'un ensemble de programmes légers implantés au niveau de chaque nœud du réseau qui ont pour but l'analyse des fichiers afin d'en extraire les séquences d'appels systèmes. Le système est aussi composé de plusieurs moniteurs où chaque moniteur prend en charge une portion du réseau et reçoit les appels systèmes extraits par les programmes chargés de l'analyse. Chaque moniteur est doté d'un mécanisme d'anti-obscurcissement qui est basé sur une méthode d'alignement d'appels systèmes, ainsi qu'une méthode probabiliste pour la représentation des comportements des programmes. Tous les moniteurs identifient d'une façon collaborative les familles des malwares en partageant les informations sur les comportements des malwares via une infrastructure de partage basée rendez-vous (RENShare), à l'aide d'une table de hachage distribuée (DHT, pour Distributed Hash Table). Cette approche vient réduire la dépendance de la totalité du réseau à une entité centrale dans une architecture centralisée. Cependant, les moniteurs restent les maillons faibles de ce type d'approches.

Fung et al. [Fung et al., 2014] ont proposé un système totalement distribué pour la détection collaborative des malwares. Le principe de celui-ci, est de faire collaborer différents outils an-

tivirus installés sur différentes machines d'un réseau afin d'améliorer la précision de détection. L'analyse s'effectue en transmettant le fichier suspect à tous les outils des machines voisines. Celles-ci, vont à leur tour analyser et transmettre le fichier a leurs voisins, le mécanisme de décision est basé sur l'ensemble de l'historique des résultats d'analyse des différents outils participant à l'analyse. Bien que ce système soit totalement distribué, il est inadéquat pour un déploiement à large échelle, vu qu'il n'envisage pas de mécanisme pour l'identification des fichiers, et se contente de transférer le fichier en entier d'un nœud à un autre, ce qui pose un problème dans le cas d'un réseau avec des ressources en bande passante limitées. Aussi, il n'est pas commode du point de vue étique de transmettre un fichier qu'on soupçonne d'être un malware, et ce quelque soit les mesures de prévention qu'on peut imaginer pour assurer la sécurité des récepteurs.

Vu l'essentiel des travaux que nous avons revu dans ce manuscrit, mais aussi ce que ressort de la littérature très abondante, il en ressort deux limitations majeures :

1) Les systèmes existants sont soit centralisés, soit hiérarchiques, et ceci pose un problème quant à leur déploiement dans un réseau large échelle, tel qu'Internet.

2) Soit, ils sont distribués, mais faits de sorte que la collaboration concerne tous les nœuds. Et de ce fait, il est également impossible d'utiliser de tels systèmes avec des réseaux WAN.

5.5 Approche proposée

Comme ça a été précédemment, notre approche pour la détection de malwares est basée sur les systèmes multi-agents (SMA), vue que la technologie agent s'adapte bien au développement des systèmes distribués et ont été largement utilisés dans le domaine de la sécurité informatique, et en particulier pour la détection des intrusions [Herrero and Corchado, 2009]. Cette grande popularité dont jouissent les SMAs dans ce domaine, est due principalement à deux points majeurs qui sont : (1) *la réduction de la charge du réseau :* en effet au lieu de transférer les données dans la totalité du réseau, ce qui peut augmenter considérablement la quantité d'information transitant par le réseau, les agents peuvent être dispatchés dans la machine ou les données résident. (2) *l'autonomie :* pour les systèmes distribués à large échelle, l'aptitude de rester opérationnel lorsqu'une portion du réseau est hors service ou isolée est importante. Cette caractéristique peut être accomplie en utilisant les agents, du fait qu'un agent peut s'accommoder en l'absence d'un ou plusieurs autres agents.

Notre objectif est donc de considérer ces avantages afin de construire un système distribué pour la détection collective et coopérative des malwares. Par ailleurs, et en considérant le cas des réseaux large échelle, on doit être en mesure de prendre des décisions, en ne prenant en compte qu'un voisinage plus ou moins large d'un nœud donné. Les SMAs fournissent le

cadre méthodologique pour pouvoir réaliser cela. Un agent dans un SMA, n'as besoin, et ne devrait pas en avoir, une représentation totale de son environnement. En revanche, les méthodes d'interaction et de coopération doivent être faite de sorte à pouvoir prendre une bonne décision ou au mois acceptable, même en l'absence de l'information globale.

5.5.1 Principe de fonctionnement

Dans notre approche, l'idée est de faire communiquer plusieurs agents entre eux, chacun représentant un nœud du réseau (machine), afin d'atteindre les objectifs pour lesquels ce système a été conçu, à savoir détecter la présence d'un code PE malveillant au sein d'une machine ou d'un ensemble de machines du réseau [Belaoued and Mazouzi, 2016b]. En effet, chaque nœud est doté d'un agent situé qui possède trois rôles et qui sont l'analyse, la propagation, et la décision. L'agent dispose ainsi d'un certain nombre d'informations lui permettant de calculer son degré de certitude global d_g, telles que les degrés de certitude locaux dénotés dl_n obtenus à partir des agents situés au niveau des nœuds voisins, la signature S_f qui va permettre d'identifier d'une manière unique le fichier (f) en cours d'analyse. Un degré de similarité ds_n entre cette signature et les signatures disponibles au niveau des agents voisins. Enfin, Le nombre de nœuds (nb) participant à l'analyse. Toutes ces informations sont stockées au niveau d'une table appelée table d'analyse, et qui a la structure suivante :

Signature	dl_1	ds_1	...	dl_n	ds_n	nb	d_g
S_1	0.75	0.78	...	0.93	0.84	7	0.71
S_2	0.63	0.76	...	-	-	1	-

Tableau 5.1 – Exemple d'une table d'analyse d'un agent

L'agent réagit à deux types d'événements, qui sont la réception d'une alerte en provenance de l'un de ses voisins, et la réception d'un fichier à analyser. Le premier événement déclenche la phase de réception d'alerte et de décision, et le second déclenche tout d'abord la phase d'analyse locale et ensuite la phase de réception d'alerte et de décision.

5.5.2 La phase d'analyse locale

Cette phase consiste à analyser le fichier suspect à l'aide de l'outil de détection de malwares installé au niveau de la machine. Le résultat d'analyse (R) est dit local et est de type booléen codé en binaire. Dans le cas ou $R = 1$ (fichier malicieux) l'agent doit systématiquement propager cette information sous forme d'alerte à ses nœuds voisins (phase de réception d'alerte et de décision). Cependant, l'agent doit tout d'abord commencer par générer la signature

numérique du fichier afin de faciliter son identification par les autres nœuds du réseau. Ensuite, l'agent définir le degré de certitude de son outil d'analyse de malwares. Ces deux étapes sont présentées dans ce qui suit.

Génération de la signature (S_f) du fichier analysé

Il existe plusieurs méthodes pour générer la signature numérique d'un fichier. L'utilisation d'une fonction de hachage est parmi les méthodes les plus répondues. *Une fonction de hachage est une fonction mathématique qui permet de générer une empreinte unique d'une donnée initiale donnée en entrée* [Kizza, 2015]. L'algorithme MD5 (pour Message Digest-5) et SHA-1 (Secure Hash Algorithm 1) sont les plus utilisés pour l'identification des codes malicieux [Sikorski and Honig, 2012, Calvet, 2013]. Les deux algorithmes sont presque identiques sauf que le MD5 génère une signature de 128 bits et le SHA-1 génère une signature de 180 bits. Cependant, dans notre cas, ce type de techniques n'est pas adéquat, vu qu'un malware peut changer de forme et ainsi avoir une signature complètement différente même si son comportement reste le même. Afin d'expliquer notre point de vu, nous avons réalisé une expérience avec deux fichiers malicieux où l'un est une variante du premier (malwares métamorphiques), les deux fichiers en question sont Zbot.aacl et Zbot.aacm. Nous avons utilisé l'algorithme de hachage MD5, pour générer les signatures des deux fichiers, et nous avons obtenu les résultats présentés dans le tableau 5.2.

Fichier	Signature MD5
Zbot.aacl	9eb6fa457757710f3bfcb00c05649533
Zbot.aacm	26ad30c1bb65a193a5f60f7e36c7f004

Tableau 5.2 – comparaison de deux signatures MD5 pour deux variantes du malware Zbot

Nous pouvons voir dans le tableau 5.2, que les signatures des deux fichiers sont totalement différentes, même s'il s'agit du même malware. De ce fait, et afin d'éviter ce genre de situation où il s'agit de malwares métamorphiques, nous proposons d'utiliser une signature générique générée à partir du code opération (Opcode) extrait à partir du code machine (X86) du fichier analysé. Ceci est fait de façon automatique par l'analyse statique du fichier PE à l'aide du script Python présenté dans le figure 5.4.

Si l'on reprend l'exemple précédent, les signatures à base d'Opcodes des malware Zbot.aacl et Zbot.aacm sont présentées dans le tableau 5.3

La signature est donc composée des noms des opcodes ainsi que leurs fréquences d'apparition dans le code. Par exemple `add(19)` signifie que l'opcode add est apparue 19 fois dans le code du malware Zbot.aacl.

```
40  ep= pe.OPTIONAL_HEADER.AddressOfEntryPoint
41  ep_ava= ep+pe.OPTIONAL_HEADER.ImageBase
42  data = pe.get_memory_mapped_image()[ep:ep+100]
43  offset =0
44  while offset < len(data) :
45   i=pydasm.get_instruction( data [offset:], pydasm.MODE_32)
46   s = str(pydasm.get_instruction_string(i, pydasm.FORMAT_INTEL, 0))
47   l1 = s.split()
48   l2.append (l1[0])
49   offset += i.length
```

FIGURE 5.4 – Portion du script python utilisé pour l'extraction des Opcodes à partir d'un fichier PE.

Fichier	Signature basée sur les Opcodes
Zbot.aacl	`add(19),sub(3),mov(2),pop(2),call(2),adc(2),int3(2),and(1),...`
Zbot.aacm	`add(23),xor(4),pop(3),push(3),mov(2),inc(2),and(1), sub(1),...`

Tableau 5.3 – Signature basée sur les Opcodes des malwares Zbot.aacl et Zbot.aacm

Définir le degré de certitude local (dl_n)

Le degré de certitude local d'un nœud (dl_n) est représenté par le degré de protection d'un outils de détection de malware (sa performance) et qui est déterminé de façon périodique en se basant sur l'évaluation proposée par l'institut indépendant spécialisé dans la sécurité informatique AV-Test [2]. Cette évaluation est effectuée suivant trois différents critères qui sont : la protection, l'influence sur le système, et l'utilisation. Dans notre cas nous n'allons prendre en considération que la protection comme critère d'évaluation en considérant une valeur normalisée comprise entre 0.5 et 1.0.

Comme mentionné précédemment, une fois (dl_n) définit et la signature (S_f) générée, l'agent déclenche la phase de réception d'alerte et de décision qui va se charger de de la prise de décision et de la propagation de l'information à ses nœuds voisins. La phase d'analyse peut être résumée dans l'algorithme 1.

Algorithm 1 $Local\_Analysis(f)$

$R \leftarrow analyse(f)$
if $(R = 1)$ **then**
  $S_f \leftarrow signature(f)$
  $block(f)$
  $Recieve\_Alert\_Decision(S_f, dl_n)$
end if

2. Disponible à : `https://www.av-test.org/fr/comparer-les-resultats-des-fabricants/`

5.5.3 La phase de réception d'alertes et de décision

Lors de la réception d'une alerte provenant du même nœud A (dans le cas ou il s'agit d'un fichier analysé) ou d'un voisin B, l'agent se met à la recherche de la signature du fichier en cours d'analyse dans sa table d'analyse. Les signatures sont représentées sous forme de deux vecteurs v et u et leur comparaison se fait par la mesure du degré de similarité ds_n à l'aide de la métrique de similarité du cosinus [Sung et al., 2004]. La similarité du cosinus calcule la valeur du cosinus de l'angle (θ) entre deux vecteurs, et qui varie entre 0.0 et 1.0. Le degré de similarité ds_n est proportionnel à cette valeur, et sera d'une grande importance lors du calcul de la décision globale. Après le calcule de (ds_n), il sera comparé à un seuil α à partir duquel on peut conclure qu'on a affaire à un même fichier. Dans notre cas, nous allons considérer un seuil $\alpha = 0.75$. La similarité du cosinus est calculée à l'aide de la formule suivante :

$$ds(S_f, S_{f'}) = \cos(v, u) = \frac{v.u}{\|v\|\|u\|} = \frac{\sum_{i=1}^{n} v_i u_i}{\sqrt{\sum_{i=1}^{n} (v_i)^2}\sqrt{\sum_{i=1}^{n} (u_i)^2}} \tag{5.1}$$

Dans le but de tester l'efficacité et la robustesse du modèle de signature proposé, ainsi que la mesure de similarité employée, nous avons effectué deux expérimentations. La première consiste à évaluer notre modèle de signature dans le cas de la similarité intra-famille (la sensibilité), c.-à-d, le potentiel qu'a notre signature à représenter les malwares d'une même famille. Pour cela, nous avons calculé le degré de similarité entre les signatures de malwares de trois familles différentes. La deuxième expérimentation consiste à évaluer notre modèle de signature dans le cas de la similarité inter-famille (la spécificité), c.-à-d, l'aptitude qu'a notre signature à différencier entre deux malwares de familles différentes. Les résultats de cette expérimentation sont présentés dans les tableaux 5.4 et 5.5.

Malware	Degré similarité (ds_n)
Bagle.a Bagle.aav	0.79
Bifrose.sln Bifrose.smt	1.00
Zbot.aacl Zbot.aacm	0.96

Tableau 5.4 – Comparaison intra-famille du degré de similarité des signatures

Nous avons obtenu des résultats très satisfaisants avec une similarité intra-famille (sensibilité) élevée et une similarité inter-famille (spécificité) relativement faible. De ce fait, on peut conclure que notre méthode de signature sera adéquate pour le nommage des fichiers sujets à une analyse. Nous avons aussi réalisé une expérimentation en considérons non pas

Malware	Degré similarité (ds_n)
Bifrose.sln Zbot.aacl	0.14
Bagle.a Zbot.aacl	0.23
Bagle.aav Bifrose.sln	0.25

Tableau 5.5 – Comparaison inter-famille du degré de similarité des signatures

la liste d'Opcodes comme signatures mais la liste des APIs (Application Programming Interface) importées par le fichier analysé. Cependant, nous avons obtenus des résultats moins satisfaisants en matière de similarité intra-famille comparés à ceux obtenus avec les signatures à base d'opcodes. Par exemple, on a obtenu une similarité de 0.14 pour les malwares Bagle.a et Bagle.aav, et une similarité de 0.55 pour Zbot.aacl et Zbot.aacm. C'est pour cette raison que nous n'avons opté, que pour le modèle de signatures à base d'Opcodes.

Après le calcule du degré de similarité (ds_n), et s'il existe une correspondance entre la signature reçue dans l'alerte et l'une des signatures enregistrées dans la table d'analyse ($ds_n > 0.75$), l'agent comptabilise le nombre de nœuds participant à cette analyse (nb). Si (nb) est supérieur à un certain seuil θ que nous fixons à 50% des nœuds voisins au nœud A, l'agent calcule une décision globale (dg) à l'aide de la formule suivante :

$$d_g = \frac{1}{V^A} \sum dl_n \times ds_n \tag{5.2}$$

Où :

- V^A représente le nombre de nœuds voisins au nœud A.
- dl_n est le degré de certitude local d'un nœud n voisin à A.
- ds_n est le degré de similarité entre les signatures du fichier analysé.

Une fois (dg) calculé, il est comparé à un seuil $\beta = 0.75$. Si dg est supérieur à ce seuil, alors, l'agent conclut qu'il s'agit bien d'un malware, et met la valeur de dg à 1 afin d'influencer les décisions des nœuds voisins, ensuite il communique sa décision à l'aide de la méthode $Send\_Alert()$ (voir l'algorithme 2). Celle-ci invoque la méthode $Recieve\_Alert\_Decision()$ de l'agent V_i avec comme paramètres, g, et la signature s_f du fichier malicieux. Enfin, l'agent va se remettre dans un état d'attente. Dans le cas où il n'y a aucune correspondance de la signature du fichier analysé avec les signatures disponibles au niveau de la table d'analyse

($ds \leq 0.75$), l'agent ajoute cette signature à sa table avec le (dl_n) reçu et propage l'alerte à ses voisins à l'aide de la méthode $Send\_alert()$ avec comme paramètres le dl_n et s_f reçus. Une fois la communication de l'alerte terminée, l'agent se met dans un état d'attente jusqu'à la réception d'un nouveau fichier à analyser ou d'un événement qui va déclencher la phase de réception d'alertes et de décision.

Algorithm 2 $Send\_Alert(S_f, d, V_i)$

$@V_i.Recieve\_Alert\_Decision(S_f, d)$

La phase de réception d'alerte et de décision a été définit sous forme de méthode nommée $Recieve\_Alert\_Decision()$ et qui est présentée dans l'algorithme 3.

5.6 Discussion

La décision consensuelle au sein du réseau, est matérialisée par le calcul et la propagation du degré d'alerte global (dg). Ce dernier permet d'une part, de réduire les alertes correspondant aux faux positifs quand les nœuds concernés ne sont pas secondés par d'autres qui affirment ces alertes. D'autre part, les nœuds, dont les outils de protection sont peu performants ou obsolètes, vont être protégés en exploitant les alertes de sécurité en provenance des nœuds bien protégés du réseau, et ce via leurs nœuds voisins respectifs.

Le modèle de signature proposé est robuste contre les techniques d'obscurcissement citées précédemment. En effet, le fait de ne prendre en compte que les opérandes, a permis de contourner systématiquement la technique de rennomage de registres. Par ailleurs, le fait de considérer l'occurrence de chaque opérande va nous permettre de contourner la technique d'obscurcissement à base réagencement d'instructions, vu que nous ne considérons pas l'ordre des Opcodes.

Par ailleurs, notre métrique est globalement résistante à l'obscurcissement par insertion de code poubelle. En effet, si le code est inséré en dehors des sections de code, il n'aura aucun effet, étant donné que nous considérons que le code qui est à l'intérieur de ces sections.

Algorithm 3 La méthode *Recieve_Alert_Decision*(S_f, dl_n)

```
for i =1, SizeOF(Analyse_Table) do
    if match (Analyse_Table[i].Signature, S_f) then
        match ← true
        Analyse_Table[i].nb++
        Analyse_Table[i].dl_n ← dl_n
        Analyse_Table[i].ds_n ← ds_n
        if (Analyse_Table[i].nb > δ) then
            Calculate(d_g)
            if (d_g > β) then
                d_g ← 1
                remove(f)
                for all agents (V_i) neighbooring (A) do
                    if V_i ≠ V_n then
                        Send_Alert(S_f, d_g, V_i)
                    end if
                end for
                break()
            else
                d_g ← 0
                unblock(f)
                break()
            end if
        end if
    end if
end for
if match = false then
    Analyse_Table[i+1].Signature ← S_f
    Analyse_Table[i+1].dl_n ← dl_n
    Analyse_Table[i+1].nb++
    for all agents (V_i) neighbooring (A) do
        if V_i ≠ V_n then
            Send_Alert(S_f, dl_n, V_i)
        end if
    end for
end if
```

Par contre, si le code est inséré à l'intérieur des sections de code, nous montrons que l'ajout d'instructions machines, selon les mêmes proportions que celles dans le code d'origine (plausible si les tailles du code et du code inséré sont suffisamment élevées), ne change pas significativement le degré de similarité (ds).

Soit :

v, et u les deux vecteurs selon la formule 5.1, et soit u' le nouveau vecteur correspondant à l'insertion dans u d'un code (signature $S_{f'}$). Si on suppose que les codes-opérations restent uniformément distribués dans le PE correspondant à u', qui peut exprimer ceci comme suit : $\forall i, u'_i = \lambda u_i$.

d'où :

$$ds(S_f, S_{f''}) = \cos(v, u') = \frac{v.u'}{\|v\|\|u'\|} = \frac{\sum_{i=1}^{n} v_i \lambda u_i}{\sqrt{\sum_{i=1}^{n} (v_i)^2}\sqrt{\sum_{i=1}^{n} (\lambda u_i)^2}} \tag{5.3}$$

$$ds(S_f, S_{f''}) = \cos(v, u') = \frac{v.u'}{\|v\|\|u'\|} = \frac{\lambda \sum_{i=1}^{n} v_i u_i}{\sqrt{\sum_{i=1}^{n} (v_i)^2}\lambda\sqrt{\sum_{i=1}^{n} (u_i)^2}} \tag{5.4}$$

$$ds(S_f, S_{f''}) = \cos(v, u') = \frac{v.u'}{\|v\|\|u'\|} = \frac{v.u}{\|v\|\|u\|} = ds((S_f, S_{f'}) \tag{5.5}$$

Enfin, dans le cas où il s'agit d'un obscurcissement basé sur le remplacement d'instructions équivalentes, nous suggérons d'utiliser une table de correspondance, qui va contenir toutes les équivalences d'instructions. Le calcul de la similarité prendra en compte toutes les combinaisons possibles. Cela est envisageable vu qu'il existe un nombre restreint de codes-opérations, cependant, ce cas précis n'a pas été traité dans cette thèse.

5.7 Conclusion

Dans ce dernier chapitre, nous avons présenté une approche distribuée à base d'agents pour la détection collective et coopérative des malwares. La motivation qui nous a poussé à proposer une approche collaborative, dans un contexte distribué, est la multitude et l'hétérogénéité des outils locaux de détection.

Notre contribution consiste dans cette partie de la thèse à la mise en place des rôles des agents ainsi que le mécanisme de communication entre les différents agents, ce qui va

permettre leur collaboration afin d'aboutir à une décision consensuelle sur la présence ou non d'une attaque par malwares. Dans notre cas, nous préconisons l'utilisation des méthodes statistiques, telles que celles présentées au chapitre précédent, pour la catégorisation de code. En effet, ce sont des méthodes temps-réel, ce qui est souhaitable dans ce genre de domaine.

Comme travaux futurs, Nous projetons d'améliorer notre algorithme de génération de signatures en y incluant un mécanisme de correspondance des Opcodes, qui va rendre le modèle de signature robuste quant aux techniques d'obscurcissement basées sur le remplacement d'instructions équivalentes. Par ailleurs, nous projetons de proposer notre propre méthode d'évaluation des outils anti-malware en les évaluant sur un nombre important d'échantillons de malwares. En dernier lieu, nous allons procéder à l'implémentation de notre système, ainsi qu'à son évaluation sur un réseau réel, type WAN.

Chapitre 6

Conclusion générale

Sommaire

Selon l'institut spécialisé en sécurité informatique AV-TEST [1], il existe à ce jour (début 2016) plus de 400 millions de malwares qui circulent sur Internet. Et avec plus de 300 mille nouveaux malwares découverts chaque jour. Ce nombre devrait doubler dans quelques années. La guerre qui oppose, d'une part, les créateurs de malwares qui ont pour objectif de créer des malwares de plus en plus difficile à détecter, et d'une autre part, les chercheurs en sécurité informatique qui veulent que les systèmes informatiques soient des plus hermétiques ainsi que de développer des systèmes de détection de malwares de plus en plus efficaces, ne va pas cesser d'aussi tôt. Et nul ne peut prédire qui en sortira vainqueur, vu qu'il n'existe ni arme, ni protection absolue.

6.1 Résumé des contributions

Dans ce qui suit nous allons proposer un petit récapitulatif de notre travail, qui consistait au développement d'un système temps réel pour la détection de malwares, et en la proposition d'une approche pour la détection collective et coopérative des malwares.

En premier lieu, nous avons présenté les différents concepts liés au domaine de la sécurité informatique d'une manière générale. Ensuite, nous avons mis l'accent sur les attaques par malwares, ainsi que les mécanismes de protection contre ce type d'attaques. Notre contribution à travers cette thèse se résume donc en trois points principaux, et qui peuvent se présenter comme suit :

Notre premier objectif, était de proposer un système temps-réel pour la détection des malwares, basé sur les tests statistiques d'hypothèses. Un grand pas vers cet objectif a été fait, vu que notre système a permis de détecter 100% des malwares analysés avec un taux de

1. `http://www.av-test.org/en/statistics/malware/`

faux positif de moins de 5%, et un temps moyen de détection égal à 0.09 secondes par fichier PE analysé.

Deuxièmement, nous avons présenté une nouvelle approche basée sur la méthode de l'analyse des correspondances multiples (ACM) pour l'identification des différentes associations d'APIs utilisées par les différents types de malwares (virus, vers, chevaux de Troie, etc.). Ceci nous permettra de distinguer efficacement entre les différentes catégories de malwares ayant été analysées.

Comme dernière contribution, nous avons proposé un mécanisme de décision collective et collaborative pour la détection des malwares. Il s'agit d'une approche basée SMA dont le but est de permettre la collaboration des différents outils de détection de malwares présents dans un ensemble de machines reparties sur un réseau large échelle.

6.2 Travaux futurs et perspectives

Comme travaux futures, nous projetons d'améliorer notre système de détection local de malwares en y introduisant de nouveaux attributs tels que les instructions machines et d'autres informations structurelles. Ceci va permettre d'augmenter la précision de détection. Une autre perspective concernant notre système de détection local qui consiste à intégrer la notion d'apprentissage incrémental, qui va permettre au système d'acquérir de nouveaux exemples d'entrainement au fur et à mesure de son exécution. Notre dernière perspective concerne notre approche proposée pour la détection collective des malwares, où nous projetons de l'implémenter et l'expérimenter un réseau WAN réel.

Au-delà de la détection des malwares, qui devrait continuer, et de se consolider tant qu'il y aura des Hackers qui exploitent des failles de code pour s'incursire dans le système, il est toujours question de faire l'effort pour le développement sain et sécurisé des applications. Ce n'est qu'en assurant des applications totalement dépourvues de failles de sécurités, qu'on peut espérer stopper en grande partie le développement de malwares.

Un aspect aussi important, et qu'on devrait prévoir pour l'éradication des malwares, consiste en la vulnérabilité humaine. Il est également question de mettre l'accent sur la formation en sécurité informatique pour tous les utilisateurs, et aussi pour les développeur.

Bibliographie

[Abbott et al., 1976] Abbott, R. P., Chin, J. S., Donnelley, J. E., Konigsford, W. L., Tokubo, S., and Webb, D. A. (1976). Security analysis and enhancements of computer operating systems. Technical report, DTIC Document.

[Alam et al., 2014] Alam, S., Traore, I., and Sogukpinar, I. (2014). Current trends and the future of metamorphic malware detection. In *Proceedings of the 7th International Conference on Security of Information and Networks*, page 411. ACM.

[Alazab et al., 2014] Alazab, M., Huda, S., Abawajy, J., Islam, R., Yearwood, J., Venkataraman, S., and Broadhurst, R. (2014). A hybrid wrapper-filter approach for malware detection. *Journal of Networks*, 9(11) :2878–2891.

[Aycock, 2006] Aycock, J. (2006). *Computer viruses and malware*, volume 22. Springer Science & Business Media.

[Ballmann, 2015] Ballmann, B. (2015). *Understanding Network Hacks*. Springer.

[Bazrafshan et al., 2013] Bazrafshan, Z., Hashemi, H., Fard, S. M. H., and Hamzeh, A. (2013). A survey on heuristic malware detection techniques. In *Information and Knowledge Technology (IKT), 2013 5th Conference on*, pages 113–120.

[Belaoued and Mazouzi, 2014] Belaoued, M. and Mazouzi, S. (2014). Statistical study of imported apis by pe type malware. In *Advanced Networking Distributed Systems and Applications (INDS), 2014 International Conference on*, pages 82–86. IEEE.

[Belaoued and Mazouzi, 2015a] Belaoued, M. and Mazouzi, S. (2015a). An mca based method for api association extraction for pe malware categorization. *International Journal of Information and Electronics Engineering*, 5(3) :225.

[Belaoued and Mazouzi, 2015b] Belaoued, M. and Mazouzi, S. (2015b). A real-time pe-malware detection system based on chi-square test and pe-file features. In *Computer Science and Its Applications*, pages 416–425. Springer.

[Belaoued and Mazouzi, 2015c] Belaoued, M. and Mazouzi, S. (2015c). Towards an automatic method for api association extraction for pe-malware categorization. In *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication*, page 40. ACM.

[Belaoued and Mazouzi, 2016a] Belaoued, M. and Mazouzi, S. (a). Chi-square based decision for real-time malware detection using pe-file features. *Journal of Information Processing Systems [In Pres]*.

[Belaoued and Mazouzi, 2016b] Belaoued, M. and Mazouzi, S. (b). Vers une approche collective et collaborative pour la détection des malwares. In *Colloque sur l'Optimisation et les systèmes d'Information (COSI'2016), Submitted*.

[Belaoued et al., 2015] Belaoued, M., Mazouzi, S., Noureddine, S., and Salah, B. (2015). Using chi-square test and heuristic search for detecting metamorphic malware. In *New Technologies of Information and Communication (NTIC), 2015 First International Conference on*, pages 1–4. IEEE.

[Bisbey and Hollingworth, 1978] Bisbey, R. and Hollingworth, D. (1978). Protection analysis : Final report. *University of Southern California Information Sciences Institute, Tech. Rep*.

[Bishop, 1999] Bishop, M. (1999). Vulnerabilities analysis. In *Proceedings of the Recent Advances in Intrusion Detection*, pages 125–136.

[Bond, 1990] Bond, A. H. (1990). Distributed decision making in organizations. In *Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on*, pages 896–901. IEEE.

[Boudaoud, 2002] Boudaoud, K. (2002). *Détection d'intrusions : Une nouvelle approche par systèmes multi-agents*. PhD thesis, Université de Versailles, France.

[Brand, 2010] Brand, M. (2010). *Analysis avoidance techniques of malicious software*. PhD thesis, Edith Cowan University.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1) :5–32.

[Briffaut, 2007] Briffaut, J. (2007). *Formalisation et garantie de propriétés de sécurité système : application à la détection d'intrusions*. PhD thesis, PhD thesis, Université d'Orléans. SDS.

[Bye et al., 2010] Bye, R., Camtepe, S. A., and Albayrak, S. (2010). Collaborative intrusion detection framework : Characteristics, adversarial opportunities and countermeasures. In *CollSec*.

[Calvet, 2013] Calvet, J. (2013). *Analyse dynamique de logiciels malveillants*. PhD thesis, École Polytechnique de Montréal.

[Chaib-Draa et al., 2001] Chaib-Draa, B., Jarras, I., and Moulin, B. (2001). Systèmes multi-agents : principes généraux et applications. *Edition Hermès*, pages 1030–1044.

[Chakradeo et al., 2013] Chakradeo, S., Reaves, B., Traynor, P., and Enck, W. (2013). Mast : triage for market-scale mobile malware analysis. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 13–24. ACM.

[Chedzoy, 2006] Chedzoy, O. (2006). Phi-coefficient. inencyclopedia of statistical sciences.

[Chien and Ször, 2002] Chien, E. and Ször, P. (2002). Blended attacks exploits, vulnerabilities and buffer-overflow techniques in computer viruses. *Virus*, 1.

[Chouaib, 2011] Chouaib, H. (2011). *Sélection de caractéristiques : méthodes et applications*. PhD thesis, Université Paris Descartes.

[Cohen, 1987] Cohen, F. (1987). Computer viruses : theory and experiments. *Computers & security*, 6(1) :22–35.

[Ding et al., 2013] Ding, Y., Yuan, X., Tang, K., Xiao, X., and Zhang, Y. (2013). A fast malware detection algorithm based on objective-oriented association mining. *computers & security*, 39 :315–324.

[Drogoul, 1993] Drogoul, A. (1993). *De la simulation multi-agent à la résolution collective de problèmes*. PhD thesis, Université Paris VI.

[Du et al., 2015] Du, P., Samat, A., Waske, B., Liu, S., and Li, Z. (2015). Random forest and rotation forest for fully polarized sar image classification using polarimetric and spatial features. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105 :38–53.

[Dua and Du, 2011] Dua, S. and Du, X. (2011). *Data mining and machine learning in cybersecurity*. CRC press.

[Eilam, 2011] Eilam, E. (2011). *Reversing : secrets of reverse engineering*. John Wiley & Sons.

[Erbschloe, 2004] Erbschloe, M. (2004). *Trojans, worms, and spyware : a computer security professional's guide to malicious code*. Butterworth-Heinemann.

[Farrington and Loeber, 1989] Farrington, D. P. and Loeber, R. (1989). Relative improvement over chance (rioc) and phi as measures of predictive efficiency and strength of association in 2× 2 tables. *Journal of Quantitative Criminology*, 5(3) :201–213.

[Ferber, 1995] Ferber, J. (1995). Les systèmes multi-agents, vers une intelligence collective. *InterEditions, Paris*.

[Ferber, 1997] Ferber, J. (1997). Les systèmes multi-agents : un aperçu général. *Techniques et sciences informatiques*, 16(8).

[Ferber, 1999] Ferber, J. (1999). *Multi-agent systems : an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading.

[Filiol, 2009] Filiol, É. (2009). *Les virus informatiques : théorie, pratique et applications*. Springer Science & Business Media.

[Fung et al., 2014] Fung, C. J., Lam, D. Y., and Boutaba, R. (2014). Revmatch : An efficient and robust decision model for collaborative malware detection. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–9. IEEE.

[Fürnkranz et al., 2012] Fürnkranz, J., Gamberger, D., and Lavrač, N. (2012). *Foundations of rule learning*. Springer Science & Business Media.

[Gadelrab, 2008] Gadelrab, M. E.-S. (2008). *Évaluation des Systèmes de Détection d'Intrusion*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

[Galal et al., 2015] Galal, H. S., Mahdy, Y. B., and Atiea, M. A. (2015). Behavior-based features model for malware detection. *Journal of Computer Virology and Hacking Techniques*, pages 1–9.

[Gazet, 2010] Gazet, A. (2010). Comparative analysis of various ransomware virii. *Journal in computer virology*, 6(1) :77–90.

[Guessoum, 2003] Guessoum, Z. (2003). Modèles et architectures d'agents et de systèmes multi-agents adaptatifs. *Dossier d'habilitation à diriger des recherches de l'Université Pierre et Marie Curie*.

[Hald et al., 1952] Hald, A. et al. (1952). Statistical tables and formulas.

[Halfond et al., 2006] Halfond, W. G., Viegas, J., and Orso, A. (2006). A classification of sql-injection attacks and countermeasures. In *Proceedings of the IEEE International Symposium on Secure Software Engineering*, volume 1, pages 13–15. IEEE.

[Han and Lee, 2009] Han, S.-W. and Lee, S.-J. (2009). Packed pe file detection for malware forensics. *The KIPS Transactions : PartC*, 16(5) :555–562.

[Hansman and Hunt, 2005] Hansman, S. and Hunt, R. (2005). A taxonomy of network and computer attacks. *Computers & Security*, 24(1) :31–43.

[Harrison, 2009] Harrison, D. M. (2009). The uncertainty in physical measurements. *American Journal of Physics*, 77(9) :862–863.

[Hauser, 2013] Hauser, C. (2013). A basis for instrusion detection in distributed systems using kernel-level data tainting.

[Herrero and Corchado, 2009] Herrero, Á. and Corchado, E. (2009). Multiagent systems for network intrusion detection : A review. In *Computational Intelligence in Security for Information Systems*, pages 143–154. Springer.

[Hiet, 2008] Hiet, G. (2008). *Détection d'intrusions paramétrée par la politique de sécurité grâce au contrôle collaboratif des flux d'informations au sein du système d'exploitation et des applications : mise en œuvre sous Linux pour les programmes Java*. PhD thesis, Université Rennes 1.

[Highland, 1997] Highland, H. J. (1997). A history of computer viruses : Three special viruses. *Computers & Security*, 16(5) :430–438.

[Howard, 1997] Howard, J. D. (1997). An analysis of security incidents on the internet 1989-1995. Technical report, DTIC Document.

[Jacob et al., 2008] Jacob, G., Debar, H., and Filiol, E. (2008). Behavioral detection of malware : from a survey towards an established taxonomy. *Journal in computer Virology*, 4(3) :251–266.

[Jacob et al., 2009] Jacob, G., Debar, H., and Filiol, E. (2009). Malware detection using attribute-automata to parse abstract behavioral descriptions. *arXiv preprint arXiv :0902.0322*.

[Jang-Jaccard and Nepal, 2014] Jang-Jaccard, J. and Nepal, S. (2014). A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*, 80(5) :973–993.

[Jensen, 2002] Jensen, R. S. (2002). *Immune system for virus detection and elimination*. PhD thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark.

[June, 2010] June, I. (2010). Anti-malware vendors slow to respond. *Computer Fraud & Security*, pages 1–2.

[Kephart et al., 1997] Kephart, J. O., Sorkin, G. B., Chess, D. M., and White, S. R. (1997). Fighting computer viruses : Biological metaphors offer insights into many aspects of computer viruses and can inspire defenses against them. *Scientific American*.

[Kern et al., 2007] Kern, C., Kesavan, A., and Daswani, N. (2007). *Foundations of security : what every programmer needs to know*. Apress.

[Kizza, 2015] Kizza, J. M. (2015). Guide to computer network security. *Computer Communications*.

[Kramer and Bradfield, 2010] Kramer, S. and Bradfield, J. C. (2010). A general definition of malware. *Journal in computer virology*, 6(2) :105–114.

[Kumar, 1995] Kumar, S. (1995). *Classification and detection of computer intrusions*. PhD thesis, Purdue University.

[Kuncheva and Rodríguez, 2007] Kuncheva, L. I. and Rodríguez, J. J. (2007). An experimental study on rotation forest ensembles. In *Multiple Classifier Systems*, pages 459–468. Springer.

[Labbe, 2005] Labbe, K. G. (2005). Evaluation of two host-based intrusion prevention systems. Technical report, DTIC Document.

[Le Bars, 2003] Le Bars, M. (2003). *Un Simulateur Multi-Agent pour l'Aide à la Décision d'un Collectif : Application à la Gestion d'une ressource Limitée Agro-environnementale*. PhD thesis, Université Paris Dauphine-Paris IX.

[Le Roux and Rouanet, 2010] Le Roux, B. and Rouanet, H. (2010). *Multiple correspondence analysis*, volume 163. Sage.

[LeDoux and Lakhotia, 2015] LeDoux, C. and Lakhotia, A. (2015). Malware and machine learning. In *Intelligent Methods for Cyber Warfare*, pages 1–42. Springer.

[Li et al., 2011] Li, X., Loh, P. K., and Tan, F. (2011). Mechanisms of polymorphic and metamorphic viruses. In *Intelligence and Security Informatics Conference (EISIC), 2011 European*, pages 149–154. IEEE.

[Lu et al., 2013] Lu, H., Wang, X., Zhao, B., Wang, F., and Su, J. (2013). Endmal : An anti-obfuscation and collaborative malware detection system using syscall sequences. *Mathematical and Computer Modelling*, 58(5) :1140–1154.

[Ludwig, 1998] Ludwig, M. A. (1998). *The giant black book of computer viruses*. American Eagle Publications.

[Lutz, 2013] Lutz, M. (2013). *Learning python*. " O'Reilly Media, Inc.".

[Malin et al., 2011] Malin, C. H., Aquilina, J. M., and Casey, E. (2011). *Malware Forensics Field Guide for Windows Systems : Digital Forensics Field Guides*. Elsevier.

[Marschalek et al., 2014] Marschalek, M., Kimayong, P., and Gong, F. (2014). Cyphort Labs Special Report POS Malware Revisited : Look What We Found Inside Your Cashdesk. Technical report, Cyphort labs.

[Mazouzi, 2008] Mazouzi, S. (2008). *RECONNAISSANCE DE FORMES PAR LES SYSTEMES AUTO-ORGANISES : Application aux images de profondeur*. PhD thesis, UNIVERSITÉ MENTOURI DE CONSTANTINE.

[Mitnick and Simon, 2004] Mitnick, K. D. and Simon, W. L. (2004). *L'art de la supercherie : l'importance du facteur humain dans la sécurité informatique : les révélations du plus célèbre hacker de la planète*. CampusPress.

[Nafir et al., 2014] Nafir, A., Mazouzi, S., and Chikhi, S. (2014). Collective intrusion detection in wide area networks. In *Innovations in Intelligent Systems and Applications (INISTA) Proceedings, 2014 IEEE International Symposium on*, pages 46–51. IEEE.

[Natani and Vidyarthi, 2013] Natani, P. and Vidyarthi, D. (2013). Malware detection using api function frequency with ensemble based classifier. In *Security in Computing and Communications*, pages 378–388. Springer.

[Nazario, 2004] Nazario, J. (2004). *Defense and detection strategies against Internet worms*. Artech House.

[Oberheide et al., 2008] Oberheide, J., Cooke, E., and Jahanian, F. (2008). Cloudav : N-version antivirus in the network cloud. In *USENIX Security Symposium*, pages 91–106.

[Okamoto and Ishida, 2000] Okamoto, T. and Ishida, Y. (2000). A distributed approach against computer viruses inspired by the immune system. *IEICE transactions on communications*, 83(5) :908–915.

[O'Kane et al., 2011] O'Kane, P., Sezer, S., and McLaughlin, K. (2011). Obfuscation : The hidden malware. *Security & Privacy, IEEE*, 9(5) :41–47.

[Parker, 1998] Parker, D. B. (1998). *Fighting computer crime : A new framework for protecting information*. John Wiley & Sons, Inc.

[Pieprzyk et al., 2013] Pieprzyk, J., Hardjono, T., and Seberry, J. (2013). *Fundamentals of computer security*. Springer Science & Business Media.

[Pietrek, 1994] Pietrek, M. (1994). Peering inside the pe : a tour of the win32 (r) portable executable file format. *Microsoft Systems Journal-US Edition*, pages 15–38.

[Pilou, 2006] Pilou, J.-F. (2006). Tout sur les systèmes d'information.

[Potter and Day, 2009] Potter, B. and Day, G. (2009). The effectiveness of anti-malware tools. *Computer Fraud & Security*, 2009(3) :12–13.

[Rajesh et al., 2015] Rajesh, B., Reddy, Y. J., and Reddy, B. D. K. (2015). A survey paper on malicous computer worms. *International Journal of Advanced Research in Computer Science and Technology*, 3.

[Rish, 2001] Rish, I. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York.

[Rodriguez et al., 2006] Rodriguez, J. J., Kuncheva, L. I., and Alonso, C. J. (2006). Rotation forest : A new classifier ensemble method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10) :1619–1630.

[Salehi et al., 2014] Salehi, Z., Sami, A., and Ghiasi, M. (2014). Using feature generation from api calls for malware detection. *Computer Fraud & Security*, 2014(9) :9–18.

[Sang, 2012] Sang, F. L. (2012). *Protection des systèmes informatiques contre les attaques par entrées-sorties*. PhD thesis, INSA de Toulouse.

[Schultz et al., 2001] Schultz, M. G., Eskin, E., Zadok, E., and Stolfo, S. J. (2001). Data mining methods for detection of new malicious executables. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 38–49. IEEE.

[Shabtai et al., 2009] Shabtai, A., Moskovitch, R., Elovici, Y., and Glezer, C. (2009). Detection of malicious code by applying machine learning classifiers on static features : A state-of-the-art survey. *Information Security Technical Report*, 14(1) :16–29.

[Shafiq et al., 2009] Shafiq, M. Z., Tabish, S. M., Mirza, F., and Farooq, M. (2009). Peminer : Mining structural information to detect malicious executables in realtime. In *Recent advances in intrusion detection*, pages 121–141. Springer.

[Shankarpani et al., 2012] Shankarpani, M., Kancherla, K., Movva, R., and Mukkamala, S. (2012). Computational intelligent techniques and similarity measures for malware classification. In *Computational Intelligence for Privacy and Security*, pages 215–236. Springer.

[Shen et al., 2002] Shen, Y.-D., Zhang, Z., and Yang, Q. (2002). Objective-oriented utility-based association mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 426–433. IEEE.

[Siddiqui et al., 2008a] Siddiqui, M., Wang, M. C., and Lee, J. (2008a). A survey of data mining techniques for malware detection using file features. In *Proceedings of the 46th Annual Southeast Regional Conference on XX*, ACM-SE 46, pages 509–510, New York, NY, USA. ACM.

[Siddiqui et al., 2008b] Siddiqui, M., Wang, M. C., and Lee, J. (2008b). A survey of data mining techniques for malware detection using file features. In *Proceedings of the 46th annual southeast regional conference on xx*, pages 509–510. ACM.

[Sikorski and Honig, 2012] Sikorski, M. and Honig, A. (2012). *Practical malware analysis : the hands-on guide to dissecting malicious software*. no starch press.

[Silva et al., 2010] Silva, C., Sousa, P., and Veríssimo, P. (2010). Rave : Replicated antivirus engine. In *Dependable Systems and Networks Workshops (DSN-W), 2010 International Conference on*, pages 170–175. IEEE.

[Simonin and Ferber, 2003] Simonin, O. and Ferber, J. (2003). Un modèle multi-agent de résolution collective de problèmes situés multi-échelles. *Technique et Science Informatiques*, 22(4) :317–329.

[Singh, 2009] Singh, A. (2009). *Identifying Malicious Code Through Reverse Engineering*, volume 44. Springer Science & Business Media.

[Singh and Chauhan, 2009] Singh, Y. and Chauhan, A. S. (2009). Neural networks in data mining. *Journal of Theoretical and Applied Information Technology*, 5(6) :36–42.

[Skoudis and Zeltser, 2004] Skoudis, E. and Zeltser, L. (2004). *Malware : Fighting malicious code*. Prentice Hall Professional.

[Stamp, 2011] Stamp, M. (2011). *Information security : principles and practice*. John Wiley & Sons.

[Sung et al., 2004] Sung, A. H., Xu, J., Chavez, P., and Mukkamala, S. (2004). Static analyzer of vicious executables (save). In *Computer Security Applications Conference, 2004. 20th Annual*, pages 326–334. IEEE.

[Szor, 2005] Szor, P. (2005). *The art of computer virus research and defense*. Pearson Education.

[Timm and Perez, 2010] Timm, C. and Perez, R. (2010). *Seven deadliest social network attacks*. Syngress.

[Tittel, 2004] Tittel, E. (2004). *PC magazine fighting spyware, viruses, and malware*, volume 3. John Wiley & Sons.

[Toderici and Stamp, 2013] Toderici, A. H. and Stamp, M. (2013). Chi-squared distance and metamorphic virus detection. *Journal of Computer Virology and Hacking Techniques*, 9(1) :1–14.

[Vasilomanolakis et al., 2015] Vasilomanolakis, E., Karuppayah, S., Mühlhäuser, M., and Fischer, M. (2015). Taxonomy and survey of collaborative intrusion detection. *ACM Computing Surveys (CSUR)*, 47(4) :55.

[Vinod et al., 2009] Vinod, P., Jaipur, R., Laxmi, V., and Gaur, M. (2009). Survey on malware detection methods. In *Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security (IITKHACK'09)*, pages 74–79.

[Wang et al., 2009a] Wang, C., Pang, J., Zhao, R., Fu, W., and Liu, X. (2009a). Malware detection based on suspicious behavior identification. In *Education Technology and Computer Science, 2009. ETCS'09. First International Workshop on*, volume 2, pages 198–202. IEEE.

[Wang et al., 2009b] Wang, C., Pang, J., Zhao, R., and Liu, X. (2009b). Using api sequence and bayes algorithm to detect suspicious behavior. In *Communication Software and Networks, 2009. ICCSN'09. International Conference on*, pages 544–548. IEEE.

[Wang, 2006] Wang, W. (2006). *Steal this computer book 4.0 : what they won't tell you about the Internet*. No Starch Press.

[Wassermann and Su, 2008] Wassermann, G. and Su, Z. (2008). Static detection of cross-site scripting vulnerabilities. In *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on*, pages 171–180. IEEE.

[Weaver et al., 2003] Weaver, N., Paxson, V., Staniford, S., and Cunningham, R. (2003). A taxonomy of computer worms. In *Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 11–18. ACM.

[Witten et al., 2011] Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining : Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc.

[Wooldridge et al., 1995] Wooldridge, M., Jennings, N. R., et al. (1995). Intelligent agents : Theory and practice. *Knowledge engineering review*, 10(2) :115–152.

[Wu and Banzhaf, 2010] Wu, S. X. and Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems : A review. *Applied Soft Computing*, 10(1) :1 – 35.

[Wu et al., 2003] Wu, Y.-S., Foo, B., Mei, Y., and Bagchi, S. (2003). Collaborative intrusion detection system (cids) : a framework for accurate and efficient ids. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 234–244. IEEE.

[Ye et al., 2010] Ye, Y., Li, T., Jiang, Q., and Wang, Y. (2010). Cimds : adapting postprocessing techniques of associative classification for malware detection. *Systems, Man, and Cybernetics, Part C : Applications and Reviews, IEEE Transactions on*, 40(3) :298–307.

[Ye et al., 2008] Ye, Y., Wang, D., Li, T., Ye, D., and Jiang, Q. (2008). An intelligent pe-malware detection system based on association mining. *Journal in computer virology*, 4(4) :323–334.

[Zaki and Sobh, 2004] Zaki, M. and Sobh, T. S. (2004). A cooperative agent-based model for active security systems. *Journal of network and computer applications*, 27(4) :201–220.

[Zhou et al., 2010] Zhou, C. V., Leckie, C., and Karunasekera, S. (2010). A survey of coordinated attacks and collaborative intrusion detection. *Computers & Security*, 29(1) :124–140.

[Zou et al., 2002] Zou, C. C., Gong, W., and Towsley, D. (2002). Code red worm propagation modeling and analysis. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 138–147. ACM.

[Zou et al., 2006] Zou, C. C., Towsley, D., and Gong, W. (2006). On the performance of internet worm scanning strategies. *Performance Evaluation*, 63(7) :700–723.