UQAC
Université du Québec
à Chicoutimi

MÉMOIRE

PRÉSENTÉ À

L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

BENJAMIN VIGNAU

LA SÉCURITÉ DANS L'INTERNET DES OBJETS : DES CONFIGURATIONS PAR DÉFAUT AUX DÉNIS DE SERVICES

SEPTEMBRE 2020

TABLE DES MATIÈRES

Table des matières i

Table des figures iii

Liste des tableaux v

Résumé 1

Remerciements 2

Introduction 4

1 Les objets connectés : monde de diversité 11
- 1.1 Un bref historique . . . 11
- 1.2 Les différents types d'objets connectés . . . 13
  - 1.2.1 L'architecture générale d'un système connecté et intelligent . . . 15
  - 1.2.2 Les systèmes autonomes . . . 18
  - 1.2.3 Les capteurs et effecteurs . . . 18
  - 1.2.4 Les concentrateurs . . . 19
  - 1.2.5 Les interfaces hommes-machines . . . 20
- 1.3 L'état de la sécurité des données dans le monde des objets connectés . . . 21
  - 1.3.1 Critères d'analyse . . . 23
  - 1.3.2 ZigBee . . . 28
  - 1.3.3 Z-Wave . . . 33
  - 1.3.4 Synthèse . . . 37
  - 1.3.5 Les objets directement connectés à Internet . . . 39

2 Les logiciels malveillants et les réseaux de zombies d'objets connectés 43
- 2.1 Définitions . . . 43
  - 2.1.1 Logiciel malveillant . . . 44
  - 2.1.2 Réseaux de zombies . . . 44
- 2.2 Les problèmes causés par les réseaux de zombies . . . 47
  - 2.2.1 Le déni de service . . . 48
  - 2.2.2 Le spam . . . 49
  - 2.2.3 Miner des cryptomonnaies . . . 50

2.3 Les outils d'étude . . . 51
2.3.1 Les pots de miel . . . 52
2.3.2 Les analyse réseaux . . . 53
2.4 Problématique et question de recherche . . . 55

3 Les évolutions des logiciels malveillants et de leurs réseaux de zombies 57
3.1 Méthodologie . . . 57
3.1.1 Les taxonomies des réseaux de zombies . . . 58
3.1.2 Les outils de modélisation de la propagation des réseaux de zombies . . . 59
3.2 Études des taxonomies existantes et des familles de programmes malveillants . . . 60
3.2.1 Les taxonomies existantes . . . 60
3.2.2 Organiser les divers comportements . . . 65
3.2.3 Analyse des familles de programmes malveillants . . . 66
3.3 Notre taxonomie . . . 74
3.3.1 Explication des taxons . . . 75
3.3.2 Une nouvelle représentation de l'évolution des logiciels malveillants . . . 86

4 Outils de simulation d'infection et de propagation d'épidémie 93
4.1 Les modèles de simulations d'infections de réseaux de zombies . . . 94
4.1.1 Les modèles existants . . . 94
4.1.2 Critères d'analyse des modèles existants . . . 96
4.1.3 Analyse des modèles sélectionnés . . . 97
4.1.4 Zombots : le jeu de simulation des pandémies de robots zombies . . . 99
4.2 Les simulations effectuées avec notre modèle. . . . 103
4.3 Les résultats des simulations . . . 108
4.3.1 Expérience 1A . . . 108
4.3.2 Expérience 1B . . . 112
4.3.3 Expérience 2A . . . 116
4.3.4 Expérience 2B . . . 123
4.3.5 Expérience 3A . . . 130
4.3.6 Expérience 3B . . . 134

5 Critiques et interprétation 139
5.1 Interprétations des graphes d'évolutions . . . 139
5.2 Critiques et pistes d'amélioration pour les représentations . . . 142
5.3 Interprétations des simulations de propagations d'infections . . . 143
5.4 Critiques . . . 146
5.5 De nouvelles solutions ? . . . 147
5.5.1 Prévoir les futurs avancées . . . 147
5.5.2 Les réseaux de bienveillance . . . 148

Conclusion 152

Bibliographie 157

TABLE DES FIGURES

1.1 Prévisions de l'évolution du nombre d'objets connectés sur dix ans (Lueth, 2018) 13
1.2 Évolution de l'intérêt pour les objets connectés 14
1.3 Architecture des objets connectés 17
1.4 Les couches du protocole ZigBee, tiré de Xueqi *et al.* (2017) 29

2.1 Architecture du botnet Mirai, d'après (Ji *et al.*, 2018) 45
2.2 Cycle de fonctionnement général d'un réseau de zombie grandissant 46
2.3 Cycle de fonctionnement de Mirai 48

3.1 Taxonomie des réseaux de zombies IoT 77
3.2 Graphe Phylogénique des programmes malveillants. 87
3.3 Propagation des fonctionnalités d'attaque. 89
3.4 Propagation des fonctionnalités d'infection. 90

4.1 Fonctionnement général d'un zombie lors du processus d'infection 100
4.2 Fonctionnement général du jeu 101
4.3 Évolution de la population du réseau #1 sur 1 500 tours (configuration avec 5 vers) 109
4.4 Évolution de la population du botnet #5 sur 1500 tours (configuration avec 5 vers) 109
4.5 Évolution de la population des 5 botnets sur 1500 tours (configuration avec 5 vers) 110
4.6 Évolution de la population des 2 botnets sur 1500 tours (configuration avec 2 vers) 111
4.7 Évolution de la population du botnet #1 sur 1500 tours 112
4.8 Évolution de la population du botnet #3 sur 1500 tours 113
4.9 Évolution de la population du botnet #5 sur 1500 tours 114
4.10 Évolution de la population des 5 botnets sur 1500 tours 115
4.11 Évolution de la population du botnet #1 sur 1000 tours 117
4.12 Évolution de la population du botnet #1 sur 5 000 tours 118
4.13 Évolution de la population du botnet #2 sur 1 000 tours 119
4.14 Évolution de la population du botnet #2 sur 5 000 tours 120
4.15 Évolution de la population des deux botnet sur 5 000 tours 121
4.16 Zoom de l'évolution de la population des deux botnet sur 5 000 tours 122
4.17 Évolution de la population du botnet #1 sur 1000 tours 124
4.18 Évolution de la population du botnet #1 sur 5000 tours 125
4.19 Zoom de l'évolution de la population du botnet #1 sur 5000 tours 126
4.20 Évolution de la population du botnet #2 sur 1000 tours 127
4.21 Évolution de la population du botnet #2 sur 5000 tours 128

4.22 Zoom de l'évolution de la population du botnet #2 sur 5000 tours 129
4.23 Évolution de la population des deux botnets sur 1000 tours 130
4.24 Évolution de la population du botnet #1 sur 1 500 tours 132
4.25 Évolution de la population du botnet #2 sur 1 500 tours 133
4.26 Évolution de la population des deux botnet sur 1 500 tours 134
4.27 Évolution de la population du botnet #1 sur 1500 tours 135
4.28 Évolution de la population du botnet #2 sur 1500 tours 136
4.29 Évolution de la population des deux botnet sur 1500 tours 137

LISTE DES TABLEAUX

1.1 Niveau de sécurité des protocoles ZigBee et Z-Wave 37

3.1 Comparaison des différentes taxonomies 64
3.2 Sources utilisées pour analyser les familles de réseaux de zombies (1/3) 74
3.3 Sources utilisées pour analyser les familles de réseaux de zombies (2/3) 75
3.4 Sources utilisées pour analyser les familles de réseaux de zombies (3/3) 76
3.5 Fonctionnalités des programmes malveillants et de leur réseaux de zombies . . 85

4.1 Comparaison des différents modèles 102
4.2 Expérience 1 : concurrence entre réseau du même type 106
4.3 Expérience 2a : concurrence entre stratégie de scan aléatoire et séquentielle . . 106
4.4 Expérience 2b : concurrence entre stratégie de scan aléatoire et séquentielle . . 107
4.5 Expérience 3a : concurrence, stratégie identique avec surpression 107
4.6 Expérience 3b : concurrence, stratégie différente avec surpression 107

RÉSUMÉ

Notre projet de recherche consiste à étudier divers aspects de la sécurité des objets connectés (ou IoT), et plus particulièrement leur exploitation de masse. En effet, ces dernières années, nous avons constaté la multiplication des attaques de déni de services provoqués par des réseaux de zombies d'objets connectés. Un réseau de zombies est un ensemble d'objets connectés, serveurs ou ordinateurs infectés et contrôlés à distance par un programme malicieux. Nous souhaitions donc comprendre les mécanismes permettant à des attaquants d'infecter des centaines de milliers d'objets connectés pour ensuite les coordonner et provoquer de grandes attaques de déni de services. Pour ce faire, nous avons d'abord étudié les objets connectés en général, les principales applications et les principaux mécanismes de sécurité. Nous avons analysé en détail deux des protocoles de communication les plus utilisés dans le monde des objets connectés. Ensuite, nous nous sommes concentrés sur les objets assurant le pont entre le réseau Internet et le réseau local d'objets connectés. En effet, ce sont ces objets directement connectés à Internet qui se font massivement exploiter pour former des réseaux de zombies. Nous avons étudié divers réseaux de zombies afin de lister et comprendre chacune de leurs fonctionnalités. Les buts ici sont multiples : créer une taxonomie pour réseaux de zombies d'objets connectés et dans un second temps, essayer de comprendre l'évolution de ces réseaux en étudiant l'évolution de leur fonctionnalité. Ainsi, nous avons mis en place une taxonomie comportant 46 taxons

représentant chacun une fonctionnalité. Ces taxons sont classés en plusieurs familles, basées sur le cycle de vie des réseaux de zombies. Notre taxonomie comporte 8 taxons spécifiques aux réseaux de zombies d'objets connectés. Grâce à cette taxonomie, nous avons pu mettre en place une nouvelle représentation de l'évolution de ces réseaux de zombies. Nous avons dessiné les graphes de propagation de fonctionnalités permettant de montrer à quel moment est apparue une fonctionnalité, quels sont les réseaux qui l'implémentent, etc. Enfin, pour mieux comprendre pourquoi, certaines fonctionnalités disparaissaient au profit d'autres, nous avons supposé que ces dernières devaient être plus efficaces (permettre à un réseau de zombies d'infecter plus d'objets dans un temps plus court). Nous supposons que d'autres facteurs peuvent intervenir, comme la difficulté d'implémentation de la fonctionnalité. Cependant, afin de tester notre hypothèse principale, nous avons développé un modèle de simulation d'infection. Ce modèle est inspiré par les modèles épidémiologiques utilisés en médecine afin de modéliser les propagations des maladies infectieuses au sein des populations. Cependant, au lieu d'utiliser un ensemble d'équations différentielles, nous avons développé un programme Python sous forme d'un jeu tour par tour. Cette implémentation permet d'abstraire plusieurs paramètres, comme le temps par exemple. Grâce à ce modèle, nous avons pu montrer l'efficacité d'une méthode de recherche de victime aléatoire par rapport à une méthode séquentielle. Cette large supériorité d'efficacité explique pourquoi la méthode de scan aléatoire a très vite remplacé la méthode de scan séquentielle. Nous pensons que ce modèle pourra servir plus tard, afin d'imaginer de nouvelles fonctionnalités, les tester et ainsi prévoir une partie des fonctionnalités qui apparaitront au sein des réseaux de zombies.

REMERCIEMENTS

Ces travaux de recherches ont été effectués au sein du Laboratoire d'Informatique Formelle (LIF) de l'Université du Québec à Chicoutimi sous la direction du Pr. Raphaël Khoury. Je tiens à le remercier chaleureusement pour son aide, sa disponibilité ainsi que ses précieux conseils. Je tiens aussi à remercier l'ensemble des professeurs du DIM et de l'INSA CVL m'ayant formé et aidé et soutenus au cours de ces années d'études. Mes profonds remerciements vont à ma famille et plus particulièrement à mes parents ayant relu plusieurs fois ce mémoire afin d'en améliorer le style et de corriger mes divers manquements à la langue de Molière. Enfin je souhaite remercier mes amis, de France et du Québec pour m'avoir supporté et aidé dans mes études et ma vie quotidienne. Un grand merci à tous.

INTRODUCTION

Les objets connectés, ou « Internet of Things » (IoT) en anglais, sont de plus en plus nombreux dans notre monde. On retrouve ces objets un peu partout : montres, réfrigérateurs, caméras et mêmes pantoufles connectées (Labbe, 2017). Chaque année au « Consumer Electronics Show » (CES) de Las Vegas, de nombreuses entreprises proposent leurs inventions et nous pouvons y voir des objets utilisables dans de très nombreuses situations de la vie quotidienne (Rozier, 2017).

Ces objets nous aident souvent dans nos vies quotidiennes, que ce soit en mesurant le nombre de pas faits en une journée ou en allumant une pièce dans laquelle nous entrons. Le marché français des objets connectés aurait généré plus d'un milliard d'euros de revenus en 2017 selon une étude du groupe GFK (Saint-Laurent, 2017). D'après une étude de l'IDATE (Ropert, 2017), il existerait plusieurs dizaines de milliards d'objets connectés dans le monde. Texas Instruments estime que plus de 50 milliards de ces objets seront en activité en 2020 (Chase, 2013). Nous observons donc une explosion de l'intérêt pour ces objets, ayant pour but d'être intégrés dans tous types de systèmes afin de pouvoir nous faciliter la vie au maximum ou d'augmenter le profit des entreprises (Schmeiser, 2017).

Ainsi, nous pouvons remarquer que ces nouveaux outils connectés permettent d'améliorer le

travail et la vie quotidienne de nombreuses personnes dans tous les domaines. La médecine (Farahani *et al.*, 2018), l'agriculture (Hu *et al.*, 2011), la mercatique (Schmeiser, 2017) et les villes intelligentes (Zanella *et al.*, 2014) ne sont qu'une partie des exemples.

Néanmoins, le fonctionnement normal de ces objets engendre une création importante de données personnelles. La perte ou la mauvaise utilisation de ces données peuvent amener à la création de nombreux problèmes pour les utilisateurs. Par exemple, dans le cas d'une maison intelligente, le système a besoin de capter diverses données telles que la température des pièces, l'état de tous les appareils,comme celui des ampoules (allumées ou éteintes), le niveau de luminosité à l'extérieur, etc. L'ensemble de ces données sert à améliorer le confort des utilisateurs, par exemple en allumant le chauffage quand la température chute, en allumant une ampoule lorsque l'on rentre dans une pièce, ou en fermant les volets lorsqu'il fait nuit.

Cependant, si un attaquant est capable de récupérer l'ensemble de ces données en tout temps, il pourrait être capable de déduire des informations sensibles pouvant nuire aux utilisateurs. Un des exemples serait l'analyse des variations de température ou l'allumage et l'extinction des lumières. De ces analyses, il pourrait déduire des schémas de vie des utilisateurs, savoir quand la maison est occupée, où se situent ses habitants, etc. Avec ces informations, l'attaquant pourra plus facilement organiser un cambriolage.

Dans le cas d'utilisation d'un stimulateur cardiaque, le vol et la corruption des données peuvent être fatals si l'attaquant obtient la possibilité d'envoyer une commande à l'objet pour lui faire délivrer une décharge électrique. Pire encore, si un attaquant réussit à prendre le contrôle de nombreux appareils connectés, tels que des thermostats, les conséquences peuvent être désastreuses pour nos sociétés. En effet, Soltan *et al.* (2018) ont montré qu'une augmentation brutale d'environ un pour cent de la demande énergétique lors d'un pic de demande pouvait engendrer un arrêt de quelques lignes électriques. Cela provoquerait une réaction en chaîne, stoppant ainsi toutes les lignes dépendantes des premières. Ce genre d'attaque nécessiterait la prise de contrôle

de quelques centaines de milliers d'appareils électriques à forte consommation, tels que des climatiseurs ou des chauffe-eau connectés, pour provoquer d'importantes perturbations sur le réseau d'un petit pays.

Toutefois, prendre le contrôle d'autant d'appareils simultanément peut sembler, à priori, difficile. Malheureusement, bien que le monde des objets connectés soit composé de nombreux types d'objets différents, chaque type d'objets peut se vendre en plusieurs dizaines voir centaines de milliers d'exemplaires à travers le monde. De plus, certains objets, bien que construits par différentes entreprises, peuvent posséder des failles exploitables communes. C'est particulièrement le cas dans les routeurs, des caméras connectées et des enregistreuses vidéos. Ce genre de failles, facilement exploitables et communes à plusieurs centaines de milliers d'objets connectés, ont conduit à l'apparition de réseaux de zombies, composés de routeurs, caméras, etc. L'un des plus connus, Mirai, a engendré les attaques de déni de service les plus importantes enregistrées à ce jour (Kolias *et al.*, 2017; Antonakakis *et al.*, 2017).

Ainsi, la création de techniques permettant de détecter, et contrer ce genre d'attaques malveillantes est une problématique importante. Afin de pouvoir aider la communauté scientifique à se défendre contre les réseaux de zombies d'objets connectés, nous souhaitons ici étudier l'évolution de ces réseaux de zombies. Le but est de pouvoir plus facilement les catégoriser, de mieux les comprendre, et surtout de déterminer au mieux les comportements clés de ces réseaux de zombies. Ce faisant, nous serons alors capables de déterminer les fonctionnalités les plus utilisées par ces réseaux, puis nous pourrons les étudier en profondeur afin que la communauté scientifique puisse produire des réponses adaptées à ces comportements. Ainsi nous pourrions formuler la question de recherche suivante : *comment évolue les réseaux de zombies d'objets connectés ?*.

Afin de répondre à cette problématique générale, nous allons définir et expliciter les différents types d'objets connectés puis nous analyserons les problèmes majeurs de sécurité amenant à la

création de réseaux de zombies. Pour déterminer l'évolution d'une population de réseaux de zombie, nous avons besoin d'une taxonomie permettant de différencier les comportements des réseaux de zombies. Ainsi nous avons effectué une revue de la littérature sur les taxonomies existantes et avons effectué une revue de la littérature systémique des réseaux de zombies d'objets connectés. Nous avons pu identifier 18 familles de réseaux de zombies d'objets connectés et en extraire les comportements principaux.

Grâce à cela, nous avons pu améliorer les taxonomies déjà existantes afin de mieux décrire ces programmes, leurs objectifs et leurs méthodes d'attaques. Cette taxonomie comporte 46 taxons organisés sous forme d'arbre, dont 8 taxons spécifiques aux réseaux de zombies d'objets connectés. Ces derniers n'étaient présents dans aucune autre taxonomie. Nous avons ensuite utilisé cette taxonomie afin de créer une nouvelle représentation de l'évolution des logiciels malveillants. Cette représentation utilise des graphes afin de montrer les propagations des comportements entre les réseaux de zombies.

Enfin, nous avons développé un outil de simulation de propagation des infections de réseaux de zombies. Cela nous permet de visualiser les effets que peuvent avoir diverses fonctionnalités et comportements de programmes malveillants sur leurs vitesses de propagation et donc sur leur efficacité. Cet outil nous aide *in fine* à mieux comprendre les impacts de chaque fonctionnalité et ainsi à mieux prédire leurs évolutions. Nous avons effectué quelques expériences avec cet outil afin d'étudier l'impact de quelques stratégies de recherche de victimes sur l'efficacité des réseaux de zombies. Les résultats sont intéressants et montrent que le choix de la stratégie impacte fortement l'efficacité des réseaux de zombies. Bien que les résultats soient intéressants, de nombreuses expériences peuvent encore être faites, et feront l'objet de plusieurs travaux de recherches futurs.

Ce mémoire est organisé comme suit : le premier chapitre définira les différents types d'objets connectés, retracera une partie de leur histoire et analysera brièvement le niveau général de

sécurité des principaux protocoles de communications utilisés par ces objets. Le second chapitre se concentrera sur les réseaux de zombies, leurs organisations, leurs principaux buts ainsi que les outils permettant de les étudier. Le troisième chapitre présente nos travaux concernant l'évolution des logiciels malveillants ciblant des objets connectés afin de créer des réseaux de zombies. Nous y détaillerons une nouvelle taxonomie, nos simulations de certains comportements de ces réseaux. Le quatrième chapitre sera centrée sur les interprétations et critiques de nos travaux. Nous discuterons aussi de quelques nouvelles solutions envisageables pour réduire l'impact néfaste des réseaux de zombies. Enfin nous conclurons le mémoire.

CHAPITRE 1

LES OBJETS CONNECTÉS : MONDE DE DIVERSITÉ

Nous allons expliquer tout au long de ce chapitre, les notions clés de «l'Internet des objets ». Nous allons commencer par un bref historique, qui nous permettra de bien comprendre la diversité de ces objets. Nous allons ensuite définit les concepts relatifs aux objets connectés. Enfin, nous procéderons à une analyse de sécurité de quelques-uns des protocoles de communication les plus utilisés dans ce domaine.

1.1 UN BREF HISTORIQUE

Dans cette section nous allons présenter un bref historique de la création des objets connectés, de leurs protocoles ainsi que de leur utilisation. Cet historique nous permettra par la suite de mieux appréhender l'univers des objets connectés et ainsi pouvoir en donner une définition clair.

Selon un article de Cisco (Evans, 2011), le concept d'objet connecté a été introduit par un groupe de recherche appelé « Auto-ID » vers les années 2000. Le concept mit un peu de temps avant de se développer. Ce groupe travaillait principalement sur les objets contenant des puces d'identification par radiofréquence, plus communément appelée « RFID ». Ces puces permettent de transporter un « tag » qui peut correspondre à un identifiant. Cela a été très utile pour les industries, notamment pour le transport de marchandises. En effet, grâce à cette technologie, il

est plus simple d'identifier des blocs de marchandises et de les trier automatiquement.

Cependant, les technologies ont évolué et ces objets utilisent maintenant diverses puces, qui permettent de communiquer avec divers protocoles. Ainsi, il existe aujourd'hui de nombreuses technologies de communication sans fil, toutes utilisées dans le monde de l'internet des objets. Chaque technologie possède ses propres protocoles de communication, une certaine consommation énergétique, ainsi qu'une portée différente. Certains, comme le LoraWan, permettent des communications à très longue distance (plusieurs kilomètres) à faible consommation énergétique, mais à faible débit de données. À l'opposé, le Bluetooth Low Energy (BLE) permet des communications avec de plus grands débits, aussi à faible consommations énergétique, mais à portée beaucoup plus courte (une centaine de mètres environ).

Aujourd'hui, les objets connectés se sont énormément diversifiés et englobent ainsi tous les objets, autres que les PC et les serveurs traditionnels, ayant une connection directe ou indirecte au réseau Internet. On inclut aussi tous les objets pouvant se contrôler à distance, sans forcément utiliser Internet. Par exemple, une usine peut utiliser des machines contrôlées par des opérateurs à distance, dans une salle de contrôle. Ici, l'objet n'est pas relié au réseau Internet, mais est manipulable à distance.

L'intérêt pour ces objets a énormément grandi au cours des dernières années. En effet, comme le montre un article de « IoT Analytics » (Lueth, 2018), le nombre d'objets connectés est passé de 3.8 milliards en 2015 à environ 7 milliards en 2018. De plus, comme le montre la figure 1.1, issue du même rapport, en 2018 les analystes prévoyaient une augmentation annuelle de 17% du nombre d'objets connectés, jusqu'en 2025. On peut aussi y voir une diversité dans les technologies utilisées, avec une dominance des objets contrôlés dans un réseau sans fil personnel. Ces derniers sont majoritairement utilisés en domotique.

En observant la Figure 1.2, représentant l'évolution de l'intérêt pour les recherches concernant

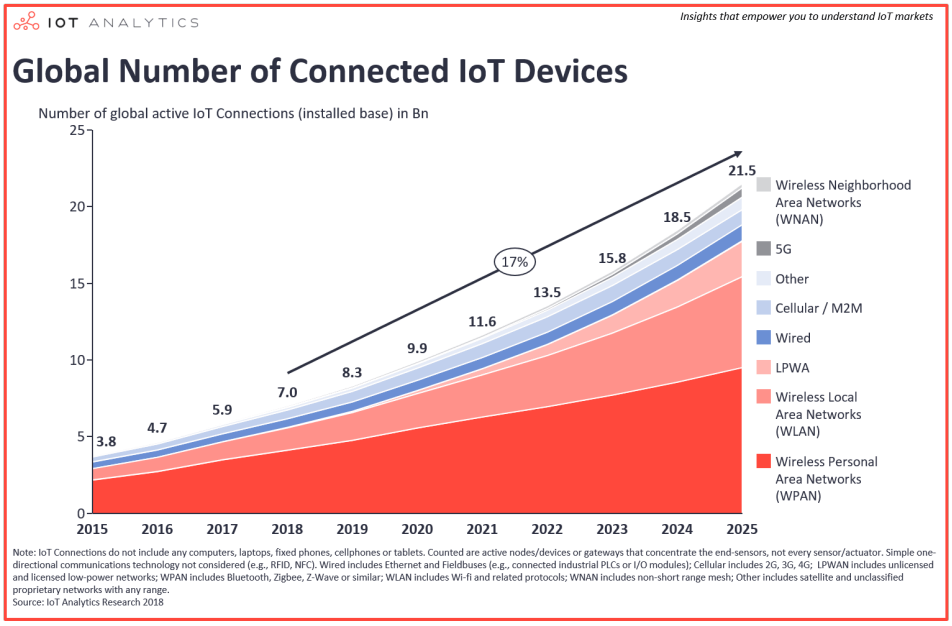

Figure 1.1 – Prévisions de l'évolution du nombre d'objets connectés sur dix ans (Lueth, 2018)

les objets connectés, on s'aperçoit que celui-ci augmente fortement à partir de 2014. On observe un phénomène similaire pour les recherches liées aux maisons intelligentes ou «smart homes". Cet intérêt pourrait s'expliquer par une amélioration suffisante des technologies afin de permettre une production plus importante d'objets connectés à coûts toujours plus faibles. Ainsi, le nombre d'applications possibles s'est fortement diversifié. Au départ, les applications étaient majoritairement tournées vers l'industrie, afin de faciliter les contrôles et transports de marchandises. Aujourd'hui, on retrouve des applications dans de nombreux domaines, comme l'agriculture, la santé, la domotique et la gestion de l'énergie.

1.2 LES DIFFÉRENTS TYPES D'OBJETS CONNECTÉS

Nous venons d'observer que le monde des objets connectés est extrêmement diversifié et hétérogène. On trouve ainsi des objets avec une faible puissance de calcul et d'autres au contraire qui possèdent un système d'exploitation complet, basé sur Linux. De ce fait, les problématiques

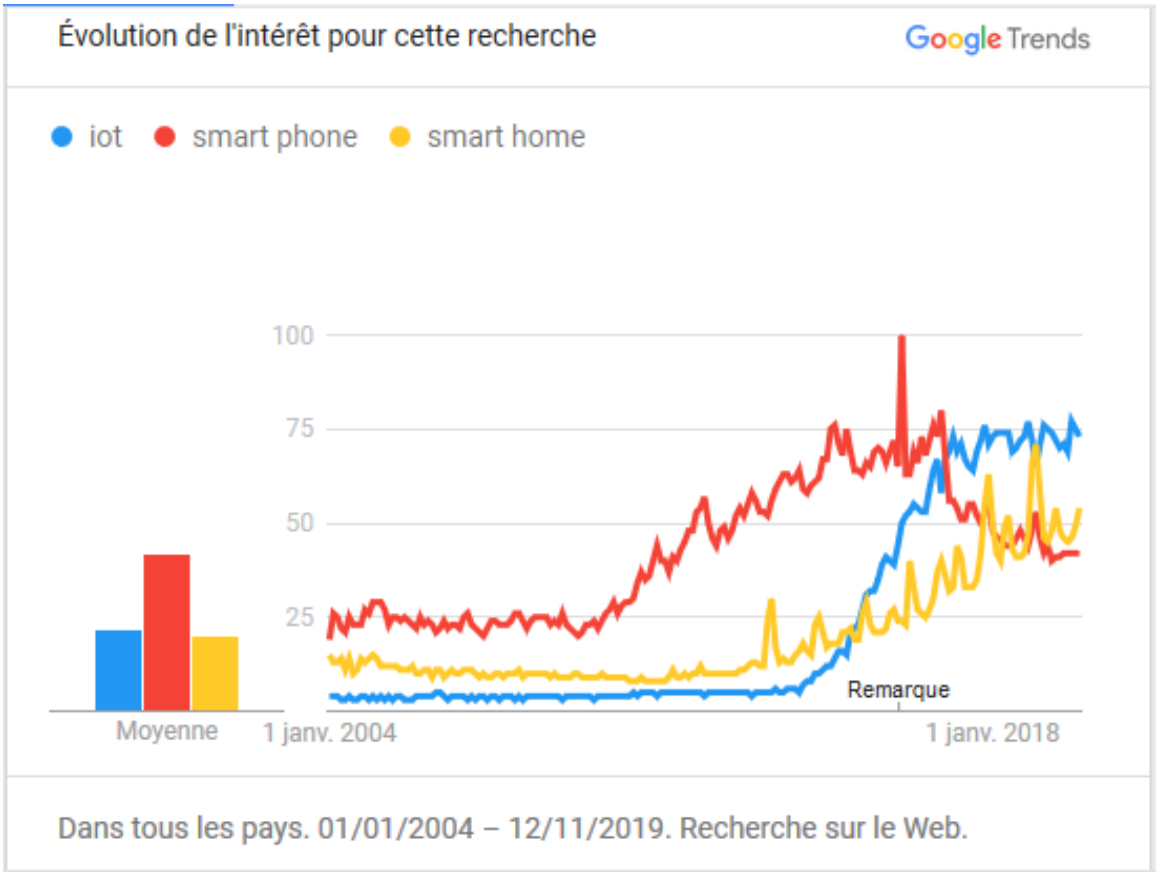

Figure 1.2 – Évolution de l'intérêt pour les objets connectés

pour ces objets peuvent être différentes et nous devons maintenant définir correctement chaque type d'objet et leurs contraintes.

1.2.1 L'ARCHITECTURE GÉNÉRALE D'UN SYSTÈME CONNECTÉ ET INTELLIGENT

Schmeiser (2017), définit les objets connectés selon les technologies utilisées, les services qu'ils doivent fournir, ainsi que leurs conditions d'utilisations. Ils sont définis ainsi :

Définition 1.1. *Les objets connectés sont des capteurs sensoriels et des acteurs (moteur etc.) utilisant des technologies de communication sans fil. Ils doivent utiliser des technologies de coopération, d'identification et de configuration distante. Ces objets doivent ainsi traiter et fournir des données sur l'environnement physique, être localisables et fournir une interface homme-machine.*

Cependant, il existe d'autres définitions comme celle de Alemdar et Ersoy (2010), qui ne parlent pas d'objets connectés mais de réseaux de capteurs sans fils.

Définition 1.2. *Un réseau de capteurs sans fils est un ensemble de capteurs et d'actionneurs communicant par un réseau sans fil (comme le Wifi, BLE etc.) afin de pouvoir récupérer des informations sur un environnement physique distant.*

On remarque que cette définition est très proche de la définition de Schmeiser (2017), et que le côté non filaire est très important. Stojkoska et Trivodaliev (2017) ont décrit les défis pour les maisons intelligentes et comment les objets connectés peuvent apporter des réponses. Ici les objets connectés sont décrits comme des objets intelligents permettant de contrôler divers équipements d'une maison, comme le chauffage ou les lumières. De cet article, nous pouvons donner la définition suivante :

Définition 1.3. *Les IoT sont une interconnexion de capteurs et d'effecteurs ayant la capacité de partager des informations au travers de plateformes logicielles communes, afin de nourrir des*

algorithmes d'analyse de données pour permettre de développer de nouvelles applications. Ces interconnexions se font souvent à l'aide de technologies de communication sans fil (RFID, WIFI etc).

De l'ensemble de ces définitions, on observe divers éléments importants : la notion de capteurs et d'effecteurs, la notion de communication et de partage de l'information, la notion de distance, de contrôle et de traitement. Le but des objets connectés est de fournir un service, ou une application exploitable par un humain. Ainsi, les interfaces personnes machines sont un élément principal des objets connectés, car elles permettent à l'humain de bénéficier du service rendu par les objets, et permettent de contrôler ces derniers. Dans ce document, nous généralisons la notion d'objets connectés afin d'englober des systèmes plus larges. Nous définissons les objets connectés et les systèmes intelligents ainsi :

Définition 1.4. *L'ensemble des objets connectés (ou IoT) correspond à l'ensemble des objets physiques et logiciels, capables d'interagir directement ou indirectement avec le monde physique, dont les commandes et résultats de ces interactions avec le monde physique doivent être gérés par des systèmes distants numériques.*

Définition 1.5. *Les systèmes intelligents sont des sous-ensembles d'objets connectés organisés et travaillant en synergie afin de fournir un service ou un ensemble de services définis.*

Ainsi, un système intelligent et connecté utilisera et orchestrera plusieurs objets connectés, tels que des capteurs, effecteurs et interface personnes-machine afin de fournir un service. Nous pouvons résumer l'architecture d'un tel système avec la figure 1.3.

Sur cette architecture, nous pouvons constater trois grandes composantes des systèmes intelligents connectés : les capteurs et effecteurs, les concentrateurs et les interfaces hommes-machines (IHMs). La première partie contient, comme son nom l'indique, les capteurs et effecteurs qui interagissent directement sur le monde physique, en mesurant une caractéristique physique

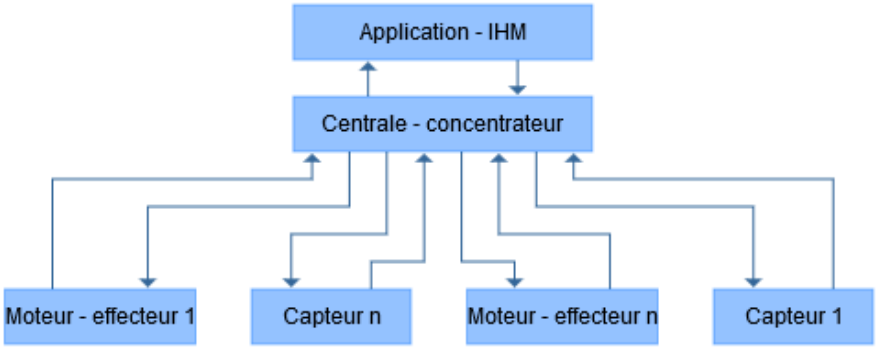

Figure 1.3 – Architecture des objets connectés

(capteur de température) ou en la modifiant (chauffage). La seconde partie contient les objets permettant de récolter et concentrer les données de plusieurs capteurs et effecteurs. C'est aussi cette partie qui comprendra les fonctions de décision et de contrôle des effecteurs.

Enfin, la troisième partie comprend les interfaces de commande qui sont des composants souvent logiciels (et rarement matériels). Ils permettent à l'utilisateur de surveiller le bon fonctionnement de ses objets et de contrôler si besoin les objets directement. C'est aussi via ces interfaces que l'utilisateur pourra changer certains paramètres du système, en indiquant une nouvelle valeur de température souhaitée par exemple.

En observant ce schéma, nous voyons que diverses entités possédant des objectifs et des ressources différentes vont devoir communiquer. En effet, un routeur ou une centrale domotique devra posséder plus de ressources qu'un capteur de température. Même au sein de ces trois familles, divers protocoles et méthodes existent.

Il est à noter qu'aucune de ces trois parties ne doit absolument être connectée à Internet. En effet, il est possible d'avoir des systèmes accessibles et manipulables uniquement depuis le réseau local.

1.2.2 LES SYSTÈMES AUTONOMES

Il existe des systèmes composés de plusieurs objets connectés, formant une entité autonome. C'est le cas par exemple des maisons connectées. Ici, nous définissons une maison connectée comme un habitat, contenant des objets connectés et travaillant en synergie afin de maximiser le confort de ses habitants. Nous appelons ce genre d'habitat des maisons intelligentes ou maisons autonomes (ou « Smart Home » ou « Home Automation » en anglais) (Bharathi *et al.*, 2017).

Dans cet exemple, le système de gestion de la maison permet de modifier divers aspects de l'environnement en fonction des données mesurées par ses capteurs. Par exemple, le système mettra en route le chauffage automatiquement s'il détecte une baisse de température ou il allumera une pièce quand il détectera une personne à l'intérieur.

On peut aussi définir les systèmes connectés intelligents comme un ensemble d'objets connectés entre eux et potentiellement au réseau Internet, travaillant en synergie afin de remplir une fonction ou un objectif en faisant intervenir le moins possible un être humain. Ainsi, une maison autonome est un système connecté intelligent. Les seules interventions de l'être humain, durant le fonctionnement nominal du système, seront de lui indiquer ses préférences (température ou taux d'humidité souhaités, etc.).

1.2.3 LES CAPTEURS ET EFFECTEURS

D'après notre définition, les capteurs physiques (tels que les capteurs de température ou d'humidité) renvoyant leurs mesures à un système distant sont des objets connectés. Il en va de même pour des effecteurs (tels que les moteurs de portails) commandés à distance. De plus, un système complexe, comprenant capteurs et effecteurs, peut aussi être considéré comme un objet connecté à partir du moment où un utilisateur distant est capable de recevoir les données mesurées par le

capteur ou s'il peut contrôler à distance ce système.

De manière générale, ces capteurs et effecteurs sont des objets simples, n'embarquant avec eux qu'un simple circuit pour transmettre leurs informations sur un réseau donné. Dans ces cas-là, ces objets n'utilisent pas un système d'exploitation complet comme un Linux Busybox, mais ils utilisent un microprogramme, aussi appelé «firmware". Ce dernier leur permet de faire l'ensemble des actions pour lesquels ils ont été prévus et rien d'autre. Un capteur de température connecté ne fera rien d'autre que relever périodiquement cette valeur physique, pour la transmettre à un concentrateur de données.

De plus, on remarquera que la majeure partie de ces objets possède une puissance de calcul plus faible que d'autres équipements tels que des téléphones intelligents ou des PC. Ceci est dû principalement à des contraintes énergétiques et économiques. Les constructeurs veulent que les objets soient fonctionnels et coûtent le moins cher possible à fabriquer. Les utilisateurs eux souhaitent avoir des objets autonomes énergétiquement afin de ne pas avoir à changer ou charger les batteries régulièrement. Cette faible puissance de calcul peut engendrer des problèmes de sécurité : elle rend plus difficile d'implémenter correctement des algorithmes de chiffrement fiables, souvent gourmands en énergie.

1.2.4 LES CONCENTRATEURS

Comme expliqué précédemment, les concentrateurs de données permettent de récupérer l'ensemble des informations perçues par les capteurs. Ils permettent aussi de distribuer les ordres aux effecteurs. C'est aussi cette partie qui peut contenir l'ensemble des algorithmes de décision dans le cadre d'un système intelligent. Les concentrateurs sont, par exemple, des routeurs ou des serveurs centraux. Cette partie permet de fournir un ensemble de données à une interface personne-machine, qui peut être locale ou distante.

Les objets servant de concentrateurs possèdent souvent une puissance de calcul bien supérieure aux capteurs et effecteurs. En effet, ils peuvent utiliser un système dérivé de Linux. Bien souvent, on appelle aussi ces systèmes « firmware » alors qu'ils sont bien plus lourds et complexes que ceux des capteurs et effecteurs. Enfin, ces appareils intègrent bien souvent leur propre interface de contrôle, que ce soit par SSH, Telnet ou via une application web.

1.2.5 LES INTERFACES HOMMES-MACHINES

Le dernier élément de notre architecture est le module des interfaces de commandes. Par exemple, une interface web, permettant à un utilisateur de connaître les différentes températures des pièces de sa maison et de contrôler son chauffage, est pour nous un objet ou une composante d'un objet connecté. Ici, l'interface n'agit pas directement sur le monde physique, elle permet de contrôler les capteurs et effecteurs. Elle agit donc de manière indirecte sur le monde physique.

Ces interfaces peuvent être physiques ou purement logicielles. Dans le premier cas, on aura par exemple, un panneau d'administration tactile d'une maison. Celui-ci va afficher diverses données à l'utilisateur, qui pourra ensuite définir des seuils ou des comportements particuliers. Dans le cas d'une interface purement logicielle, nous pouvons avoir une application web, hébergée par le système intelligent ou sur un serveur externe. Celle-ci présentera les mêmes fonctionnalités que la version physique, mais sera en plus accessible depuis plusieurs appareils.

Ici, nous avons fait le choix de considérer une interface de commande comme un objet connecté à part entière, car elle permet de contrôler indirectement d'autres objets connectés. C'est en effet un composant logiciel permettant d'agir de manière indirecte sur le monde physique, en contrôlant d'autres composants physiques. Ainsi, d'un point de vue sécurité, cette interface doit aussi utiliser des mécanismes d'authentification, d'intégrité et de confidentialité. En effet, si une entité malveillante est capable de prendre le contrôle d'une interface, d'écouter ses

communications avec les concentrateurs, voire de les modifier, elle sera capable d'influencer fortement le système intelligent et connecté.

1.3 L'ÉTAT DE LA SÉCURITÉ DES DONNÉES DANS LE MONDE DES OBJETS CONNECTÉS

Nous venons d'observer que le but de ces systèmes intelligents est de récupérer un maximum de données afin de nous offrir différents services. Par exemple, une maison intelligente va récupérer des données de température, d'humidité, d'éclairage, de qualité de l'air, afin de fournir un confort à l'utilisateur. La maison va pouvoir agir sur ces variables en allumant le chauffage ou la climatisation, en allumant ou éteignant des lumières etc.

Il en est de même pour tous les objets connectés : ils mesurent une variable physique, soit pour nous transmettre l'information, soit pour la modifier via un système intelligent. Ainsi, par nature les systèmes connectés intelligents sont amenés à collecter et utiliser de nombreuses données personnelles. De ce fait, il nous paraît nécessaire de s'intéresser à la sécurité de ces données personnelles afin qu'elles ne soient pas utilisées de manière préjudiciable à l'utilisateur final.

L'ensemble de ces données peut être détourné de son utilisation de base et une personne mal intentionnée peut provoquer de lourds dommages en prenant le contrôle d'un tel système. Nous allons maintenant analyser quelques-uns des protocoles de communications utilisés dans le domaine des objets connectés. Pour ce faire, nous étudierons les spécifications des protocoles ainsi que des articles faisant une analyse détaillée pour l'un ou l'autre des protocoles. Notre but ici est d'analyser ce que peuvent apporter ces différentes solutions en matière de sécurité générale et de protection des données personnelles.

Les protocoles que nous analysons permettent principalement la communication entre capteurs, effecteurs et concentrateurs. Ainsi, des failles de sécurité dans ces protocoles permettraient

d'attaquer un système connecté de l'intérieur ou en étant proche du système (quelques dizaines de mètres tout au plus).

Deux des protocoles les plus connus et les plus utilisés, gérant la couche physique, sont ZigBee, et Z-Wave (Salman et Jain, 2015). Ces derniers ont tous subi diverses modifications depuis leur première version, permettant d'améliorer leur efficacité et leur sécurité.

Cependant, comme le montrent certaines études, des vulnérabilités subsistent. Celebucki *et al.* (2018) ont effectué une analyse de sécurité de ces trois protocoles. Ils démontrent que des vulnérabilités subsistent, notamment durant les périodes d'échanges des clés. Ils mettent aussi en évidence que certaines de ces failles sont exploitables et peuvent mener à des pertes de confidentialité ou à des attaques de déni de service.

D'après leur étude, il serait possible de s'interposer dans une communication utilisant le protocole ZigBee, en récupérant les clés au moment de l'échange. Pour Z-Wave, le problème majeur vient de la rétrocompatibilité. Si un appareil des générations précédentes se trouve sur le réseau, alors toutes les communications se feront avec le protocole de la génération la plus ancienne. Or, les anciennes générations de ce protocole sont très mal sécurisées.

En plus des problèmes précédemment décrits, d'autres interviennent indépendamment des protocoles utilisés. En effet, afin d'avoir des coûts de production les plus faibles possible, les constructeurs d'objets connectés ont bien souvent fait des coupures au niveau de la sécurité des données.

On pourra citer par exemple l'ampoule Philips Hue pour laquelle une équipe de chercheurs a réussi à récupérer la clé de signature des mises à jour de micrologiciels. Le protocole de signature utilisé est un dérivé de l'AES. Ainsi, l'équipe de Ronen *et al.* (2017) a récupéré la clé et s'en est servi pour signer des versions malveillantes de micrologiciels qu'ils ont pu installer à distance sur des ampoules.

1.3.1 CRITÈRES D'ANALYSE

Pour cette analyse, nous allons définir trois niveaux de sécurité : -1, 0 et 1. Un niveau -1 correspond à une absence de méthode ou à l'utilisation d'une méthode obscure, inconnue ou non vérifiée. En effet, la sécurité par l'obscurité contredit les principes de Kerkhoffs (Kerckhoffs, 1883), notamment celui de la conception ouverte. La sécurité par l'obscurité est déconseillée par de nombreux experts tels que Schneier (1995). Un niveau 0 correspond à l'utilisation d'une méthode dépréciée par un organisme de standardisation ou de sécurité, tel que l'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) ou le « National Institute of Standards and Technology » (NIST). L'utilisation d'une méthode pour laquelle il existe une attaque permettant de la briser sera aussi considérée comme un niveau 0. Enfin, l'utilisation d'une méthode recommandée, mais avec un ensemble de paramètres non recommandés correspond aussi à un niveau 0. Enfin, l'attribution d'un niveau 1 de sécurité correspond à l'utilisation d'une méthode recommandée ou jugée comme acceptable par un organisme, tout en utilisant les bons paramètres. Ici, nous étudierons les propriété suivantes :

1. intégrité des communications
2. authentification des messages
3. authentification des appareils
4. impact de la rétrocompatibilité

Pour chacune des trois premières propriétés, nous utiliserons la méthode de classification en trois niveaux, comme indiqué précédemment. Pour l'impact sur la rétro compatibilité, nous ne utiliserons deux niveaux : impact fort et impact faible. Ces derniers seront décrits dans la sous-section correspondante.

Chaque propriété aura ainsi sa propre note. Cette hiérarchie en plusieurs niveaux permet de rendre compte des risques qu'engendrent les diverses méthodes utilisées. En effet, l'absence de

méthodes permettant d'assurer une des propriétés choisies rend particulièrement vulnérable un objet. Il est plus facile de l'attaquer et de le compromettre si aucun mécanisme de défense n'est mis en place.

Ici, nous voyons la sécurité des données comme un jeu, dont l'objectif serait de protéger un trésor. Si nous enterrons le trésor dans une forêt et que personne ne le surveille, alors dès qu'une personne fouillera un peu la forêt, elle trouvera notre trésor. Si maintenant, nous construisons un coffre avec une serrure et que nous faisons garder le coffre par des gardes, il sera plus difficile de trouver et de voler notre trésor. Enfin, si nous construisons un château fort, en haut d'une montagne, avec des douves, des canons et une armée pour surveiller la salle du trésor et défendre le château, il sera extrêmement difficile de dérober le trésor.

Notre analogie montre aussi qu'il faut mettre plus de moyens pour mieux sécuriser le trésor. En effet, plus l'on ajoute de couches de sécurité complémentaires, et plus il devient difficile pour un attaquant de venir dérober notre précieux trésor. On comprend donc pourquoi un objectif strict de baisse des coûts de production n'est pas compatible avec un système sécurisé. Cette analogie décrit le principe de défense en profondeur, là aussi vivement recommandé par les experts comme Schneier. "Cependant, nous pouvons aussi remarquer que le système de sécurité doit être en adéquation avec le trésor à protéger. En effet, si le système de sécurité coûte plus cher à mettre en place et à entretenir que notre trésor, alors ce système de sécurité ne sera pas viable.

Confidentialité des communications

Nous avons vu précédemment qu'un des impacts néfastes des attaques contre les objets connectés est le vol de données sensibles de l'utilisateur. Afin de réduire ce problème, il faut que les systèmes assurent une propriété de confidentialité des données. Cela se traduit par la mise en

place de méthodes afin de rendre incompréhensibles et inexploitables les données de l'utilisateur pour toute personne non autorisée. Le constructeur doit donc définir et implémenter la ou les méthodes assurant la confidentialité des données. Ainsi, si un attaquant capture les ondes émises par les objets connectés, via une technique de radio logicielle [1] par exemple (Tuttlebee, 2003), il ne pourra ni les comprendre ni les exploiter.

Intégrité des communications

Afin d'éviter qu'un attaquant puisse modifier les messages transmis par les objets connectés, il nous faut une propriété d'intégrité. Une méthode assurant l'intégrité des communications permet de détecter l'altération d'un message lors de son transport et ainsi de ne pas le prendre en compte. Cela permet d'éviter qu'un attaquant puisse modifier les messages d'un objet et ainsi donner de fausses informations au système.

Le non-respect de cette propriété pourrait, par exemple, amener un attaquant à intercepter les messages d'une serrure connectée, puis à les modifier afin que la centrale reçoive en permanence le message "la porte est fermée". Ce faisant, il pourra alors rentrer dans la maison sans déclencher l'alarme d'ouverture de porte. Ici aussi, les techniques cryptographiques sont très efficaces pour assurer cette propriété.

Authentification des messages

Afin d'éviter qu'un attaquant puisse envoyer toute sorte de messages sur le réseau, il faut implémenter une méthode d'authentification des messages. Elle permet au système de toujours savoir de quel objet provient le message, de supprimer les messages frauduleux et d'éviter

1. Technique permettant de programmer et d'utiliser n'importe quels types récepteurs et émetteurs logiciels sur un matériel générique. Ici le matériel ne fait que numériser des ondes radio, c'est le logiciel à part entière qui les interprète.

certaines injections de messages dans le réseau. L'utilisation d'un protocole ne présentant pas cette caractéristique permettra à un attaquant donner des ordres à des objets, en se faisant passer pour la centrale.

Par exemple, dans un tel cas de figure, on pourrait avoir une porte de garage connectée, s'ouvrant à la réception d'un signal particulier, que seul l'utilisateur devrait pouvoir envoyer. Dans ce cas de figure, si l'attaquant se poste à quelques mètres de distance avec une radio logicielle, il pourra alors capturer le signal et le rejouer, lui permettant ainsi d'ouvrir la porte.

Authentification des appareils

Un autre point important est la manière dont sont identifiés les appareils qui ont le droit de se connecter au réseau. En effet, il existe plusieurs méthodes et, dans le cas où il n'y aurait aucune authentification, tout appareil effectuant une requête de connexion se verrait accepté. Cela peut poser problème, car un attaquant pourrait alors connecter ses propres appareils malveillants sur le réseau de l'utilisateur. Il pourrait alors lancer d'autres attaques plus facilement et, selon les configurations du réseau outrepasser certaines méthodes de protection comme le chiffrement du réseau. C'est possible si l'ensemble du réseau utilise la même clé de chiffrement pour l'ensemble des communications. Pouvant se joindre au réseau sans approbation de l'utilisateur, l'objet de l'attaquant recevra la clé du réseau et il pourra alors s'en servir pour déchiffrer toutes les communications.

Impact de la rétrocompatibilité

Notre dernier critère d'analyse concerne la rétrocompatibilité. Cette fonctionnalité est souvent obligatoire ou fortement recommandée. Elle permet à des utilisateurs de pouvoir acheter de nouveaux équipements et de les installer chez eux, sans pour autant avoir à changer tous leurs

équipements d'origine. Cependant, la présence d'une telle fonctionnalité peut engendrer de lourds problèmes de sécurité. En effet, pour beaucoup de protocoles, les premières versions n'utilisaient aucune méthode de protection. Ainsi, il n'y avait aucune confidentialité, aucune authentification.

De ce fait, si nous avons un réseau contenant uniquement des objets neufs, utilisant les versions sécurisées des protocoles et que nous décidons d'y ajouter un équipement plus ancien, ne possédant qu'une version non sécurisée du protocole, deux cas se présentent à nous. Dans le premier, le réseau va créer un canal de communication dédié à cet objet, permettant ainsi de l'utiliser sans mettre en péril la sécurité du réseau. On va considérer ce cas comme un impact faible de la rétrocompatibilité.

Dans le second cas, le système va décider d'utiliser la même version du protocole pour communiquer avec l'ensemble des objets. De ce fait, le système va utiliser la version non sécurisée du protocole, qui est la seule disponible et utilisable par tous les objets. On va ainsi voir une baisse du niveau de sécurité de l'ensemble du réseau. On qualifiera donc cela comme un impact fort de la rétrocompatibilité.

On voit ici qu'il vaut mieux que notre protocole ait un impact faible de rétrocompatibilité. Il permet d'introduire moins de vulnérabilités et donc d'augmenter la sécurité globale de notre système intelligent. Bien entendu, une meilleure sécurité permet d'augmenter la difficulté d'une attaque contre ce système et donc de réduire les possibilités de compromission de la totalité du système. Nous allons donc attribuer un niveau -1 pour les protocoles qui ont un impact fort et un niveau 1 pour ceux ayant un impact faible.

1.3.2 ZIGBEE

Nous allons étudier dans cette section le protocole ZigBee. À l'aide des deux articles choisis (Xueqi *et al.*, 2017), (Morgner *et al.*, 2017), ainsi que des spécifications techniques (Alliance, 2014), nous avons été capables de rapidement analyser le protocole selon les critères précédemment définis. Nous ferons dans un premier temps un bref historique du protocole puis nous effectuerons l'analyse de la sécurité de ce protocole.

Historique du protocole

D'après le site officiel de ZigBee [2], le protocole ZigBee a été développé par la ZigBee Alliance, regroupant plusieurs centaines d'entreprises. Cette alliance a été fondée en 2002. Les premières spécifications du protocole ZigBee 1.0 ont été ratifiées en décembre 2004. Le protocole a été créé pour permettre la communication de petits équipements personnels, tout en utilisant le moins d'énergie possible. Le second but était de bâtir un protocole et des puces l'implémentant, au tarif le plus bas. Le protocole se base sur la norme IEEE 802.15.4 (Molisch *et al.*, 2004). ZigBee permet la mise en place de réseaux maillés, c'est-à-dire que les nœuds peuvent dialoguer entre eux, sans forcément passer par une centrale. La version que nous étudions ici est la version 3.0, déployée en 2015.

Fonctionnement général du protocole

Le protocole ZigBee est organisé en couches, un peu à la manière du modèle OSI (ISO, 1978). Ainsi, on se retrouve avec un protocole en quatre couches : physique (PHY), accès médian (MAC), réseau (NWK) et application (APL). Une représentation est donnée dans la figure 1.4

2. https ://www.zigbee.org

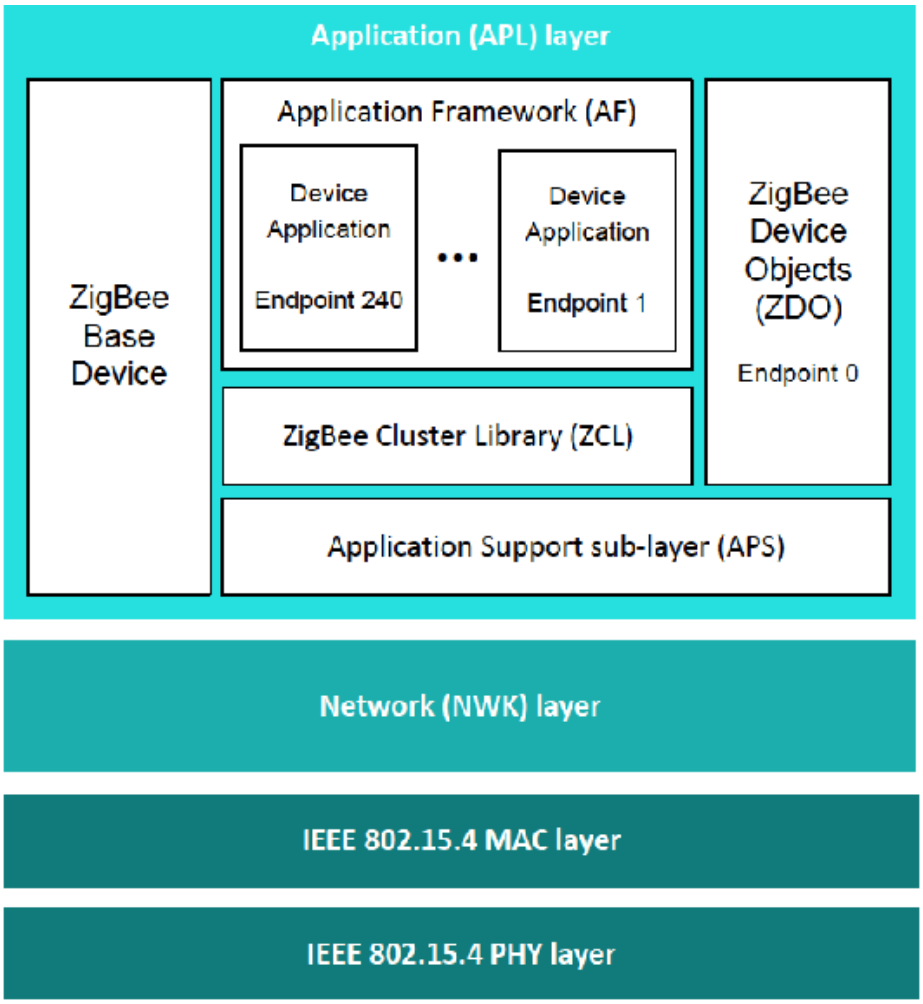

Figure 1.4 – Les couches du protocole ZigBee, tiré de Xueqi *et al.* (2017)

Ainsi, chaque couche est responsable de gérer une partie bien précise de la communication. La couche PHY est responsable de la traduction des messages en ondes physiques qui seront ensuite émises par l'appareil, puis captées et retransformées en données numériques par un autre appareil. Nous ne nous intéresserons pas à cette couche dans la suite de ce travail. Pour son fonctionnement général, ZigBee propose deux modes différents. Le premier est un mode centralisé et le second est un mode distribué. Les deux présentent des différences dans leur manière d'ajouter de nouveaux appareils dans le réseau ainsi que dans leur méthode de protection des messages. Nous voyons donc que deux de nos critères sont impactés par le mode de fonctionnement. Nous

reviendrons plus tard sur ces détails.

Analyse de sécurité

Pour commencer, il faut savoir que le protocole ZigBee assume une confiance totale entre les couches. En conséquence, les mécanismes de protection sont partagés entre les couches. D'après Fan et al. (Xueqi *et al.*, 2017), ainsi que Morgner (Morgner *et al.*, 2017), la couche MAC est définie par le protocole IEEE 802.15.4 et permet donc l'utilisation d'un algorithme de chiffrement appelé AES-CCM* (prononcé AES CCM star), utilisant une version améliorée de l'algorithme CBC-MAC faisant partie des algorithmes de code d'authentification normés par l' « International Organisation for Standardization » (ISO) ISO (2019). Il est utilisé ici avec la version AES 128, algorithme de chiffrement standard, recommandé par les grandes agences de sécurité informatique telles que l'ANSSI (ANSSI, 2014). Cet algorithme permet d'assurer les propriétés de confidentialité, d'intégrité et d'authentification des messages (Dworkin, 2004).

Cependant, il faut noter que la clé de chiffrement est partagée par tous les appareils du réseau et correspond à "la clé du réseau". Ainsi, on peut voir que si un attaquant peut connecter un équipement malveillant dans un réseau ZigBee, il obtiendra la clé de chiffrement et pourra alors compromettre les propriétés de confidentialité, d'intégrité et d'authentification des messages.

Pour la couche NWK, le principe est le même, un algorithme CCM* est utilisé, mais en utilisant les mêmes clés que pour la couche MAC. Cependant, la nouveauté de la version 3.0 de ZigBee est de pouvoir mettre en place une dernière utilisation d'un chiffrement AES-128 dans la couche APL. Dans cette dernière couche, on pourra utiliser des clés différentes de la clé de réseau afin de créer des canaux de communications sécurisés entre deux appareils. Cela peut se traduire par la mise en place d'un canal dédié entre une centrale de commande et une serrure connectée afin d'augmenter la sécurité de cette dernière (Aliance, 2017).

Maintenant que nous connaissons les mécanismes disponibles pour chaque couche, étudions les deux modes de fonctionnement du protocole afin de déterminer si ces méthodes sont correctement utilisées. D'après les équipes de recherches qui ont analysé ce protocole (Xueqi *et al.*, 2017), (Morgner *et al.*, 2017) le mode centralisé procure un plus haut niveau de sécurité. En effet, dans le mode distribué, tous les appareils doivent utiliser une clé de lien prédéfinie, pour obtenir la clé du réseau. Cette première clé, utilisée pour transmettre de manière chiffrée la clé du réseau lors de l'ajout d'un nouvel équipement, est pré-configurée par les constructeurs et devient donc la même pour tous les appareils.

Ensuite, tous les nœuds utilisent la même clé de réseau pour communiquer. Nous voyons ici une faille importante. Lors de l'appareillage, la clé du réseau est transmise à un nouvel appareil en étant chiffrée. Or comme la clé de chiffrement utilisée est connue de tous, elle devient inutile puisque tout le monde peut l'utiliser pour déchiffrer le message contenant la clé du réseau. Ainsi, dans leurs travaux, les chercheurs ont réussi à obtenir la clé du chiffrement d'un réseau ZigBee en capturant les paquets lors de l'ajout d'un nouvel appareil au réseau (Xueqi *et al.*, 2017). À partir de là, plus aucune des mesures de protection utilisées par le protocole ZigBee n'est efficace.

Concernant le mode de communication centralisé, nous pouvons observer une sécurité accrue. En effet, ce mode de communication fait intervenir un équipement particulier, appelé le centre de confiance (« Trust Center » ou TC en anglais). Cette nouvelle entité permet de centraliser la sécurité du réseau. Elle possède plusieurs fonctionnalités comme la gestion des clés et l'autorisation d'ajout de nouveaux nœuds dans le réseau. Ce dernier peut ajouter un appareil sur le réseau en utilisant une clé préconfigurée globale (la même que pour le mode décentralisé) ou une clé unique pour chaque paire d'entités. Le TC supporte aussi la mise en place de listes de contrôle d'accès (ACL) afin de déterminer à l'avance les appareils pouvant se joindre au réseau. Cette méthode d'authentification des appareils avec la méthode des clés uniques préconfigurées

est recommandée par des organismes de sécurité. On voit donc ici un plus grand niveau de sécurité, car le TC va pouvoir utiliser des clés différentes pour communiquer avec chaque nœud, permettant ainsi de conserver les propriétés de confidentialité, d'intégrité et d'authentification, même si certains appareils se sont fait corrompre. Cependant, il faut noter que cela n'est possible qu'avec des appareils compatibles dont le constructeur a correctement défini la clé unique, utilisable entre le TC et le nouveau nœud. Si cette clé n'existe pas, ce sera la clé globale, connue de tous, qui sera utilisée. C'est ce qui s'est passé dans l'expérience menée par Fan et al. (Xueqi *et al.*, 2017).

Pour finir, les spécifications de ZigBee indiquent que tous les appareils d'un même réseau doivent utiliser la même version du protocole et utiliser le même mode de communication. De plus, tous les appareils doivent être rétrocompatibles. Nous pouvons traduire cela par un impact fort de la rétrocompatibilité sur le protocole.

Outils et attaques disponibles

Afin de tester la sécurité d'un réseau ZigBee, plusieurs outils dont les sources sont libres ont été créés. Le principal est un cadriciel (ou «framework") appelé « KillerBee » [3]. Ce dernier permet d'implémenter facilement, à l'aide de composants matériels dédiés, quelques attaques simples contre les objets ZigBee. Il a d'ailleurs été utilisé par les équipes de recherches précédemment citées. Il existe aujourd'hui des surcouches pour ce logiciel, par exemple « SecBee » [4] et «Z3sec» [5]. Ces outils permettent de mettre en place diverses attaques, par exemple l'attaque « Touchlink » (Morgner *et al.*, 2017), la capture, l'analyse et le rejeu de paquets ou l'envoi massif de paquets afin de saturer le réseau (aussi appelé «flood"). L'attaque basée sur la fonctionnalité « Touchlink » permet à un attaquant d'ajouter un équipement frauduleux au sein d'un réseau

3. https ://github.com/riverloopsec/killerbee
4. https ://github.com/Cognosec/SecBee
5. https ://github.com/IoTsec/Z3sec

ZigBee. L'équipe de Morgner *et al.* (2017) montre que la présence d'un seul objet possédant cette fonctionnalité dans le réseau peut compromettre la totalité de la sécurité de ce dernier.

Enfin, certaines attaques de déni de service sont impossibles à contrer avec ZigBee. C'est le cas notamment de toutes les attaques sur le spectre ondulatoire. En effet, si un attaquant sature le spectre en utilisant une antenne de forte puissance, les objets ZigBee changeront sans cesse de canal de communication, sans jamais pouvoir se stabiliser.

Ainsi, nous pouvons conclure que la sécurité d'un réseau utilisant le protocole ZigBee dépend énormément de ses équipements et de sa configuration initiale. Un bon niveau de sécurité peut être atteint, si l'on utilise un mode de communication centralisé, avec uniquement des appareils en version 3.0 et disposant d'une clé préconfigurée unique. Il ne faut pas non plus que des objets ayant la fonctionnalité « TouchLink » soient présents sur le réseau. L'ajout d'une liste de contrôle d'accès permet aussi de mieux authentifier les appareils devant se joindre au réseau. Toute autre configuration est vulnérable.

1.3.3 Z-WAVE

Nous allons étudier dans cette section le protocole Z-Wave. À l'aide des trois articles choisis Yassein *et al.* (2018), (Badenhop *et al.*, 2017), (Rouch *et al.*, 2017) nous avons été capables de rapidement analyser le protocole selon les critères précédemment définis. Nous ferons dans un premier temps faire un bref historique du protocole puis nous effectuerons l'analyse de la sécurité de ce protocole.

Historique du protocole

Z-Wave est un protocole propriétaire, développé à partir de 2001 par la société Zensys (Ehrlich, 2008). À la base, le protocole se destinait à être principalement utilisé pour la création de

maisons intelligentes. L'entreprise fut rachetée à plusieurs reprises et la conception du protocole a mené à la création de la Z-Wave Alliance en 2005. La première version du protocole avait pour objectif de fournir de bonnes performances et ne proposait ainsi que peu de sécurité. De plus, les spécifications techniques et le code du protocole n'ont été dévoilés en partie, qu'en 2016. Avant, les études de code étaient très restreintes et les chercheurs et développeurs devaient signer des clauses de non-divulgation. Ainsi, l'étude de ce protocole fut ardue pour les équipes de chercheurs.

Fonctionnement du protocole

Tout comme le protocole ZigBee, Z-Wave est organisé en couches : la couche physique (PHY), la couche médiane (MAC), la couche adaptative contenant trois sous-couches (SAR, NWK, ENC), la couche LLC et enfin la couche applicative (APP). La couche physique sert (comme pour ZigBee) à transformer les données en signal radio. La couche MAC implémente les mécanismes d'approbation de frames, de validation de données et de retransmission. Cette couche est basée sur la norme IEEE 802.11 Yassein *et al.* (2018).

Contrairement à ZigBee, la couche MAC ne va pas fournir des méthodes pour sécuriser le protocole. La sécurité est ici faite majoritairement par la couche adaptative et notamment par la sous-couche ENC. La sous-couche réseau NWK permet de router les paquets à travers le réseau maillé, la sous-couche SAR permet de reformer les paquets qui ont été segmentés. La couche de chiffrement ENC permet de chiffrer les données avec un algorithme AES-128. Elle fournit aussi un algorithme CBC-MAC lui permettant d'assurer les propriétés de confidentialité, intégrité et authentification des communications. Nous pouvons observer que ce sont les mêmes méthodes utilisées pour assurer la sécurité du protocole ZigBee. La couche LLC permet de définir la manière dont doivent dialoguer les modules Z-Wave. Elle correspond à un jeu de commandes qui peut être envoyé et reçu. La couche application est développée par le programmeur afin de créer

un service tel que le changement de couleur d'une ampoule en fonction de la commande reçue. Ainsi, nous allons principalement analyser la couche adaptative, responsable de la sécurité du protocole.

Analyse de sécurité

Afin d'analyser la sécurité de ce protocole, les chercheurs ont dû développer diverses plateformes de radio logicielle afin de pouvoir capturer les paquets du réseau et les injecter. Cela permet aux chercheurs de se faire passer pour des utilisateurs légitimes, s'ils connaissent les clés du réseau. De plus, il faut noter que les réseaux Z-Wave sont composés de capteurs et effecteurs classiques, ainsi que d'un équipement particulier appelé «Gateway". Ce dernier permet de faire les liaisons entre les appareils, d'ajouter des nœuds dans le réseau et de lier les équipements au réseau internet. Nous apprend dans les articles (Yassein *et al.*, 2018; Badenhop *et al.*, 2017) que pour la version S2 de Z-Wave, le routeur va utiliser des clés uniques pour intégrer des équipements dans le réseau. Ces clés sont en général indiquées sur la boite de l'équipement que l'on veut intégrer. Il va ensuite y avoir un protocole d'échange de clé ECDH (Elliptic Curve Diffie Hellman), très robuste et recommandé par l'ANSSI. Cela change énormément par rapport à la précédente version qui utilisait une clé de chiffrement composée uniquement de "0" afin de chiffrer la clé du réseau. De plus, pour ajouter un nouvel élément dans le réseau il suffisait d'appuyer sur un bouton et la «gateway » ajoutait au réseau tous les équipements qui en faisaient la demande.

En fin d'année 2017, Rouch *et al.* (2017) ont réussi à créer un contrôleur universel Z-Wave, leur permettant de prendre le contrôle d'un de ces réseaux. Ici, les chercheurs capturent le «Home ID » du contrôleur originel en écoutant le réseau cible. Ensuite, ils fabriquent un faux fichier de sauvegarde et s'en servent pour forcer le contrôleur frauduleux à utiliser le «Home ID » du contrôleur sain. Ils utilisent cette méthode, car autrement il est absolument interdit

par les spécifications du protocole de choisir la valeur du «Home ID". Ensuite, les chercheurs éteignent un court moment le contrôleur sain et utilisent le contrôleur malicieux pour récupérer les connexions avec les noeuds du réseau.

Nous voyons ici une attaque intéressante, permettant aux attaquants de prendre la main sur n'importe quel réseau Z-Wave. Cependant, cette dernière requiert un accès proche du réseau cible, ainsi qu'un moyen pour redémarrer le contrôleur cible. Cela peut se faire physiquement ou en coupant la source d'énergie du bâtiment. Cette technique est donc difficile à mettre en place. De plus, dans un réseau utilisant complètement la dernière version du «secure mode", ce genre d'attaque devient impossible.

Pour finir, on apprend que les versions S0 et S2 ne sont pas compatibles et que très peu de produits possèdent la version S2 du protocole. De plus, bien que les spécifications Z-Wave S2 permettent l'utilisation de chiffrement AES et de ses dérivés, la certification n'oblige pas cette utilisation. En mai 2018, seule une quarantaine de produits étaient certifiés S2 sur le marché européen (Tierney, 2018).

En conséquence, si l'on veut utiliser un produit en version S0 dans le réseau il faut rétrograder l'ensemble de la sécurité du réseau. C'est d'ailleurs la base d'une attaque trouvée en 2018 par un laboratoire de chercheurs (Tierney, 2018). Cette dernière permet de rétrograder la sécurité d'un réseau Z-Wave S2 en une version S0. Pour ce faire, il suffit d'utiliser un objet frauduleux, faisant croire au réseau qu'il ne possède pas la version S2 du protocole. L'impact de la rétrocompatibilité est donc fort.

Contrairement au protocole ZigBee, il existe peu d'outils pour faciliter l'exploitation de réseaux Z-Wave. Seuls quelques «sniffers » physiques existent, mais ce sont les équipes d'experts qui développent leurs propres outils.

Tout comme pour le protocole ZigBee, la sécurité d'un réseau Z-Wave dépend principalement des

objets qui le composent et de la configuration mise en place par l'utilisateur. Il est possible d'avoir un réseau sécurisé à condition de n'avoir que des équipements S2, qui sont malheureusement peu nombreux. La faille nommée «Z-Shave", permettant de forcer l'ensemble du réseau Z-Wave à utiliser la version 0 (non sécurisée) reste tout de même préoccupante, car elle affecterait environ 100 millions d'appareils (cyberveille sante.gouv.fr, 2018).

1.3.4 SYNTHÈSE

Nous venons de voir que les deux protocoles analysés possèdent des configurations dans lesquelles les propriétés définies plus tôt sont respectées. Nous pouvons résumer cela dans la Table 4.1 :

Protocole	Confidentialité	Intégrité	Authentification des messages	Authentification des appareils	Impact de la rétro compatibilité
ZigBee 3.0	AES-128 Niveau 1	CCM* Niveau 1	CCM* Niveau 1	Clé pré-configurée Niveau 1	Impact Fort Niveau -1
Z-Wave S2	AES-128 Niveau 1	CBC-MAC Niveau 1	CBC-MAC Niveau 1	Clé pré-configurée Niveau 1	Impact Fort Niveau -1

Tableau 1.1 – Niveau de sécurité des protocoles ZigBee et Z-Wave

Nous pouvons donc observer que, dans les meilleures configurations, les protocoles sont globalement bien sécurisés. De plus, on observe que les deux protocoles utilisent les mêmes algorithmes pour garantir leur propriété de confidentialité et d'authentification des appareils. Leur seul point faible se trouvant au niveau de la rétrocompatibilité, qui oblige un abaissement général du niveau de sécurité si l'on souhaite utiliser d'anciens modèles. Les deux protocoles sont ainsi similaires et l'un ne surpasse pas l'autre selon nos critères.

Cependant, il faut noter que cette situation idéale n'est que peu réelle. En effet, ces dernières demandent des efforts à l'utilisateur qui se doit de paramétrer correctement le réseau. En effet, nous avons vu que la majorité des équipements Z-Wave ne possèdent pas la version sécurisée S2. De plus, le mode de communication par défaut des équipements ZigBee est le mode décentralisé pour plus de simplicité d'utilisation. Enfin, tous les articles que nous avons utilisés pour analyser ces deux protocoles mettaient en place des attaques contre ces protocoles.

Ainsi, des chercheurs ont réussi à capturer des clés de réseau ZigBee (Xueqi *et al.*, 2017) ce qui leur a ensuite permis d'espionner le réseau, de forger de faux paquets et de les injecter dans le réseau. D'autres chercheurs ont utilisé la fonctionnalité «Touchlink » de ce même protocole pour prendre le contrôle total d'un réseau ZigBee (Morgner *et al.*, 2017). Ils ont montré que la présence d'un seul équipement ayant cette fonctionnalité activée était suffisante pour mettre en péril toute la sécurité du réseau. Pour Z-Wave, les équipes ont réussi à attaquer le protocole de routage des paquets afin de créer une attaque "trou noir" (Badenhop *et al.*, 2017). Cette attaque leur permet de faire en sorte que les appareils ne puissent plus communiquer entre eux et ce sans qu'ils s'en rendent compte. Plus récemment, une équipe de chercheurs à réussi à exploiter une faille permettant de forcer un réseau Z-Wave à utiliser la version S0 du protocole, très peu sécurisé (Tierney, 2018).

1.3.5 LES OBJETS DIRECTEMENT CONNECTÉS À INTERNET

Nous venons d'analyser les deux protocoles les plus utilisés dans le monde des objets connectés et nous avons pu observer que certaines attaques sont possibles en cas de mauvaise configuration. Ceci est un véritable problème, car peu de personnes sont capables de configurer correctement leurs objets et bien souvent l'objectif des utilisateurs est d'avoir un système s'installant rapidement et facilement. Čeleda *et al.* (2010) parlent même de mentalité «plug-in-and-do-not-care » que l'on peut traduire par «branche et ne t'en occupe pas".

Cependant ces attaques ne peuvent se faire qu'à faible distance (une dizaine de mètres en général). De plus, ces objets ne sont en général pas directement connectés au réseau Internet. Ainsi, il est très difficile de mettre en place une attaque de masse, infectant rapidement des dizaines de milliers d'objets en un temps réduit. Le scénario le plus proche serait celui donné par Ronen *et al.* (2017) pour lequel les chercheurs ont récupéré la clé de signature des mises à jour du micro logiciel d'une ampoule ZigBee. Cela leur a permis de créer un programme malicieux se propageant d'ampoule en ampoule.

Cette technique est très efficace pour infecter rapidement le réseau d'un bâtiment connecté, mais dès lors que ce dernier sera éloigné de quelques dizaines de mètres des autres bâtiments, possédant eux aussi les mêmes ampoules, la propagation de l'infection cessera. Pour continuer de la propager, il faudrait plusieurs drones capables de voler proche des bâtiments, sans se faire détecter. Enfin,le ver introduit dans le code des ampoules devrait être capable de compromettre d'autres types d'objets connectés, afin de pouvoir transmettre les informations dérobées à l'attaquant. On voit ici que ce genre de scénario est peu probable, car il est extrêmement difficile à mettre en place.

C'est pourquoi s'attaquer aux concentrateurs de données ou aux interfaces de commandes parait plus efficace pour prendre le contrôle d'un système connecté. La plupart de ces systèmes sont

connectés au réseau Internet et en prendre le contrôle revient à prendre le contrôle de tout le système intelligent. De plus, il existe des objets qui cumulent l'ensemble des trois parties de notre architecture.

C'est le cas par exemple d'une caméra connectée, qui va à la fois être capteur (filmer l'environnement), concentrateur (récupération, traitement et envoi du flux vidéo) et interface de commande (interface web de configuration, lecture du flux vidéo). Ces objets sont directement connectés au réseau Internet et embarquent souvent un système d'exploitation dérivé de Linux Busybox. Il est aussi possible que ces objets soient connectés à un routeur qui fait office de passerelle vers le réseau Internet.

Un grand nombre de ces objets possèdent des configurations par défaut et demeurent accessibles directement depuis le web. Par exemple, le site Shodan [6] permet de recenser l'ensemble des objets connectés comme les routeurs, caméras et enregistreurs «streaming » . Grâce à ce site web, on peut très facilement accéder aux interfaces de commandes de ces objets et l'on observe bien souvent qu'il n'y a pas de mot de passe ou que ce dernier est très commun.

Ce manque de diversité dans les configurations de ces objets pose de grands problèmes de sécurité. En effet, il devient très facile d'accéder aux interfaces, donc aux objets et de les infecter avec du code malicieux. C'est ce qu'exploitent les entités malveillantes, depuis les années 2008-2009 pour créer des réseaux de zombies massifs. Ces réseaux sont ensuite utilisés pour lancer des attaques de déni de service distribués ou du spam. Ces attaques se répercutent sur l'ensemble des utilisateurs, qui ne peuvent pas accéder à leurs services. Ce fut le cas en 2016, avec l'attaque de Mirai contre le fournisseur de serveur de domaine Dyn (Dyn DNS, 2016). Durant plusieurs heures, les clients de la côte Est des États-Unis n'ont pas pu accéder à des services comme Amazon, Twitter ou Spotify (Wang *et al.*, 2017).

6. https ://shodan.io

Ces différents types d'attaques sont pour nous plus dangereuses, car elles peuvent se propager rapidement et de manière autonome. De plus, elles volent la puissance de calcul d'objets connectés pour créer des attaques majeures, impactant des millions d'utilisateurs à l'échelle de la planète. C'est pourquoi nous avons décidé d'étudier les programmes malveillants ciblant les objets connectés afin de créer des réseaux de zombies.

CHAPITRE 2

LES LOGICIELS MALVEILLANTS ET LES RÉSEAUX DE ZOMBIES D'OBJETS CONNECTÉS

Comme nous l'avons vu dans la partie précédente, beaucoup d'objets connectés, comme les concentrateurs, les routeurs, les caméras connectées ou les enregistreurs vidéos, utilisent un système Linux et sont vulnérables à une attaque contre les identifiants permettant de les authentifier pour utiliser l'interface d'administration. Cette vulnérabilité a été exploitée à plusieurs reprises afin de créer des réseaux de zombies. Afin de mieux comprendre les failles exploitées et ainsi, pouvoir trouver des contre-mesures adéquates, nous avons souhaité étudier les logiciels malveillants ciblant cette partie des objets connectés. Avant de décrire nos études et nos résultats, nous allons d'abord donner les définitions utiles et décrire le fonctionnement général des réseaux de zombies ainsi que les problèmes qu'ils engendrent. Nous décrirons aussi les outils les plus utilisés pour étudier ces réseaux. Enfin, nous expliciterons la problématique de recherche.

2.1 DÉFINITIONS

Nous allons ici définir les concepts de logiciels malveillants, de réseaux de zombies et nous décrirons leur organisation générale ainsi que leur cycle de vie.

2.1.1 LOGICIEL MALVEILLANT

L'équipe de Malwarebytes [1] définit un logiciel malveillant (ou « malware ») comme un terme générique qui décrit tous les programmes ou codes ayant un comportement malicieux et dommageable pour le système. Ces programmes sont hostiles et intrusifs, ils ont pour objectif de voler des données ou de stopper et d'endommager un système d'information comme un ordinateur, un réseau, une tablette ou un téléphone. Ces logiciels malveillants vont se cacher pour infecter d'autres hôtes et échapper aux systèmes de surveillance. Ces programmes peuvent aussi prendre en partie le contrôle de l'appareil infecté et donc altérer les opérations de l'utilisateur. Tout comme le virus de la grippe, les logiciels malveillants vont interférer avec le fonctionnement normal des systèmes d'information (Malwarebytes, 2019).

2.1.2 RÉSEAUX DE ZOMBIES

Les réseaux de zombies, aussi appelé « botnets » (contraction des mots « robot » et « network ») sont définis par Karspersky comme *de larges réseaux d'ordinateurs infectés par un même virus* (Fisher, 2013). Ainsi, lorsqu'un même logiciel malveillant va infecter plusieurs milliers (voir dizaines de milliers) d'objets connectés afin de les contrôler à distance, on parle de réseaux de zombies d'objets connectés. Ici, on parle de zombies du fait que les objets infectés peuvent être contrôlés en totalité par l'attaquant possédant le réseau.

Organisation et composants

Traditionnellement, on distingue deux types d'architectures pour les réseaux de zombies : centralisée ou décentralisée. La première consiste à mettre en place un serveur de contrôle et de commande (serveur C2), permettant de récupérer toutes les informations de tous les objets

1. https ://fr.malwarebytes.com/

infectés et de leur envoyer des ordres (Koroniotis *et al.*, 2019). À l'opposé, les architectures décentralisées n'ont aucun serveur central, mais utilisent un réseau pair-à-pair (« P2P ») pour communiquer. La méthode de communication centralisée a l'avantage d'être facile à implémenter et rend compte de la totalité de l'état du réseau facilement (Koroniotis *et al.*, 2019). La seconde a l'avantage d'être plus résiliente et donc plus difficile à faire tomber.

Dans le cas d'une architecture centralisée, on peut voir apparaître d'autres composants, comme le serveur de rapport ou le serveur de chargement. Ces derniers ont été introduits par Mirai (Ji *et al.*, 2018) et permettent de rendre l'architecture centralisée encore plus efficace. Le serveur de rapport permet d'enregistrer les adresses IP utilisées par des objets connectés vulnérables ainsi que leurs identifiants de connexions. Le serveur de chargement va se connecter à ces objets pour leur transmettre le logiciel malveillant. Un schéma de cette architecture est donné à la figure 2.1

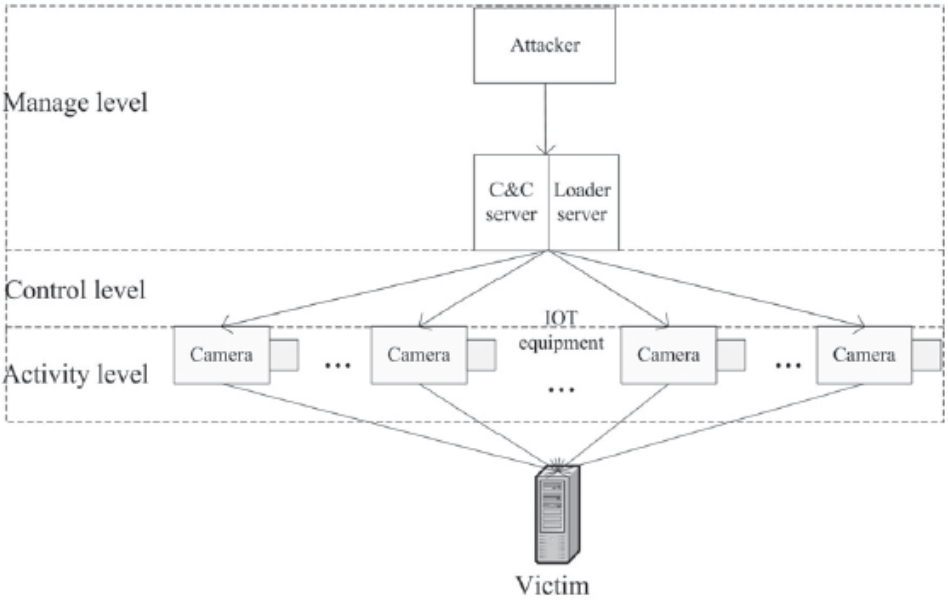

Figure 2.1 – Architecture du botnet Mirai, d'après (Ji *et al.*, 2018)

Cycle de vie

Le but principal d'un réseau de zombies est de grossir au maximum pour ensuite pouvoir lancer diverses attaques. Nous détaillerons ces dernières dans la prochaine section. Ici, nous allons décrire le fonctionnement d'un de ces réseaux, en nous basant sur le cycle de vie du programme Mirai.

Tout d'abord, il faut identifier les objets vulnérables. Pour ce faire, les réseaux de zombies doivent constamment scanner l'ensemble des adresses IP disponibles afin de trouver le maximum de victimes. Ensuite, pour chaque victime potentielle, le réseau va essayer de l'exploiter. Ensuite, si l'attaque réussit, la victime va se faire infecter et elle participera au prochain cycle. Nous pouvons résumer ce cycle en trois étapes : identification des objets, exploitation des victimes et duplication du logiciel malveillant. Un schéma récapitulatif est donné à la figure 2.2

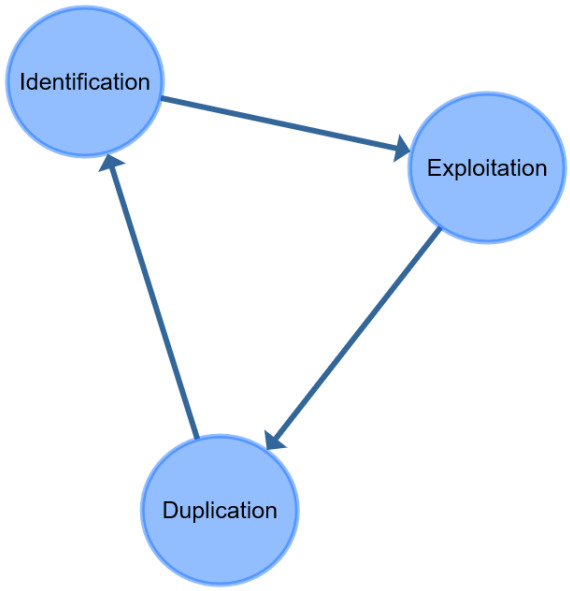

Figure 2.2 – Cycle de fonctionnement général d'un réseau de zombie grandissant

Concernant les réseaux de zombies ciblant les objets connectés, nous pouvons apporter quelques précisions sur ce fonctionnement. En effet, la phase d'exploitation est très souvent une phase

d'attaque par force brute sur les identifiants de la victime. On peut noter qu'en 2012, le réseau de zombies Carna (Carna, 2012), avait analysé l'ensemble de l'espace IP durant 24h pour créer une carte interactive des connexions au réseau Internet. Pour ce faire, il avait infecté de nombreux objets connectés qui n'avaient pas d'identifiant ou avaient des identifiants faibles. Il avait recensé plus d'un million d'objets accessibles de cette manière.

De plus, certains logiciels malveillants, comme Mirai, vont adopter une phase de dissimulation. Ils vont se charger en RAM puis supprimer leur binaire (Antonakakis *et al.*, 2017). Certains vont même changer leur nom afin d'avoir un nom de processus classique comme « telnetd » . Enfin, nous pouvons observer sur la figure 2.3 qu'une étape de rapport est mise en place. Celle-ci a été introduite par Mirai et a pour but d'améliorer la rapidité de propagation du logiciel malveillant. Cela permet à l'attaquant d'avoir une liste de tous les objets vulnérables. De plus, les zombies peuvent rapidement se remettre à scanner l'espace IP et c'est un serveur central, ayant beaucoup de puissance de calcul et de bande passante, qui s'occupe d'infecter les nouvelles victimes (Kambourakis *et al.*, 2017).

2.2 LES PROBLÈMES CAUSÉS PAR LES RÉSEAUX DE ZOMBIES

Dans cette section, nous allons détailler les principales attaques mises en place par les réseaux de zombies ainsi que leurs conséquences. D'après Jerkins (2017), les réseaux de zombies sont traditionnellement utilisés afin de créer des attaques de déni de service ou pour distribuer du « spam » (ou pourriel). Cependant, il existe aujourd'hui d'autres utilisations, par exemple, le vol d'identité, la distribution d'autres programmes malveillants ou le minage de cryptomonnaie (Koroniotis *et al.*, 2019). De manière générale, chaque réseau ne se concentre que sur un grand type d'attaque. Il existe cependant quelques exceptions, par exemple « Gameover Zeus » qui était capable de lancer des attaques de déni de services, voler des informations bancaires et lancer des campagnes de spam (Koroniotis *et al.*, 2019). Ces attaques représentent donc en

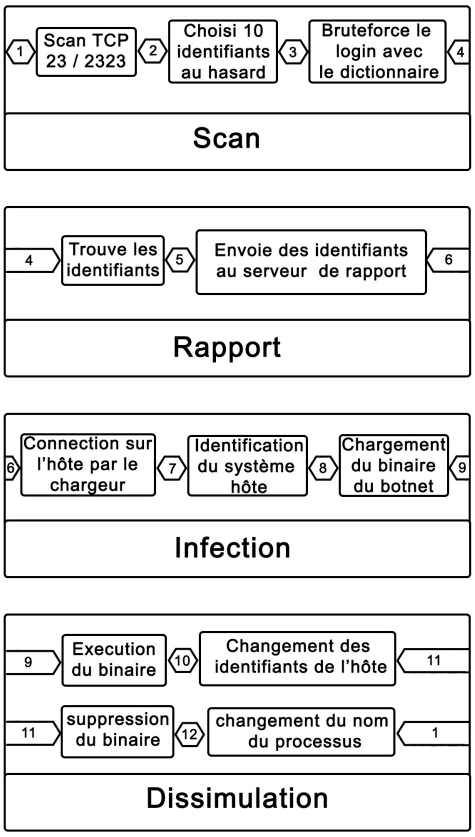

Figure 2.3 – Cycle de fonctionnement de Mirai

partie les objectifs des attaquants et en conséquences des réseaux de zombies. Nous verrons dans les chapitres 3 et 4 que ces objectifs différents peuvent amener à mettre en place des solutions différentes.

2.2.1 LE DÉNI DE SERVICE

La catégorie d'attaque la plus utilisée et la plus médiatisée est sans aucun doute l'attaque de déni de service distribuée (DDoS). Ce genre d'attaque utilise une asymétrie des ressources disponibles entre la victime et l'attaquant. Son but principal est de saturer la bande passante ou les capacités de calcul de la victime, afin que ses clients légitimes n'aient plus accès au

service. Pour mettre en place ce genre d'attaque, un grand nombre de zombies doivent utiliser au maximum leur bande passante et leur capacité de calcul afin d'envoyer le plus grand nombre possible de requêtes à la victime. On pourrait faire l'analogie avec un restaurant ou un groupe de 150 personnes viendrait occuper une salle de 100 couverts, pour demander uniquement des denrées non servies par le restaurant.

D'après Wang *et al.* (2018), le nombre et l'intensité de ces attaques étaient en croissance dans les années 2013-2014. En effet, l'équipe parle d'une augmentation de l'amplitude des attaques (taille de la bande passante utilisée) de 245% entre le premier quart de l'année 2014 et celui de 2013. À cette époque, les attaques moyennes étaient de 7.39 Gbps et la plus grosse attaque enregistrée utilisait 500 Gbps de bande passante. En 2016, les trois premières attaques du réseau Mirai étaient de 623 Gbps contre le blog de Brian Krebs et entre 1.1 et 1.2 Tbps pour les attaques contre OVH et Dyn (Antonakakis *et al.*, 2017). On voit ainsi une grande augmentation de la puissance de ces attaques au cours des dernières années.

Selon Scott Sr et Summit (2016), les réseaux de zombies sont souvent partitionnés et loués. En effet, ils montrent qu'il existe un marché appelé « DDoS-as-a-Service » qui, au lieu de proposer une application, vend des plages d'attaques de déni de service. Selon leurs travaux, le coût moyen se situe entre 25$ et 150$ par tranche de 24h contre une seule cible. Le coût varie en fonction de l'amplitude de l'attaque.

2.2.2 LE SPAM

Le concept de *spam* englobe l'ensemble des mails indésirables que l'on peut recevoir. D'après Drozhzhin (2015), ce genre de mail était à la base utilisé par les équipes commerciales, à des fins publicitaires. Cependant, les cybercriminels se sont aussi mis à utiliser les mails afin de propager leurs logiciels malveillants. En effet, c'est par le biais de pièces jointes corrompues

que se transfèrent ces programmes.

Une autre utilisation du spam par les cybercriminels est appelée hameçonnage. Cette pratique utilise l'ingénierie sociale (Kevin D.Mitnick, 2002) afin d'entraîner des usagers dans diverses arnaques. L'exemple le plus parlant est le mail d'un prince ou d'un riche homme d'affaires, vous demandant de verser de l'argent sur un compte bancaire, dans un pays différent du vôtre. Le but ici est d'utiliser les sentiments ou les émotions des victimes afin qu'elles cliquent sur la pièce jointe ou qu'elles envoient leur argent.

L'ensemble de ces techniques se base sur le fait de délivrer un nombre colossal de mails afin de toucher un maximum de personnes. Souvent, il est nécessaire qu'une petite fraction des personnes ciblées tombe dans le piège pour que l'attaquant réussisse son attaque. Par exemple, si ce dernier souhaite infecter le réseau d'une entreprise, il peut utiliser le spam afin d'envoyer une pièce jointe corrompue à l'ensemble des employés. Il n'aura besoin que d'une seule ouverture de cette pièce jointe pour infecter le réseau interne de l'entreprise. Ainsi l'utilisation de réseaux de zombies, comprenant des milliers d'ordinateurs ou d'objets connectés, facilite la distribution d'une telle quantité de courriels frauduleux.

Ici, l'avantage d'utiliser un réseau de zombies est le fait que chaque zombie enverra une fraction des mails. Ce faisant, chaque mail aura moins de chance de se faire étiqueter comme indésirable. Ce fut notamment le cas en 2014, où un réseau d'objets connectés a envoyé plus de 750 000 mails indésirables. Chaque objet n'envoyait qu'une dizaine de mails, trois fois par jour. Les chercheurs avaient même identifié un frigo parmi les zombies (Williams, 2014).

2.2.3 MINER DES CRYPTOMONNAIES

Ces dernières années, nous avons vu apparaître un engouement massif pour les cryptomonnaies. Les plus connues étant le Bitcoin (Nakamoto, 2008) et l'Etherum (Buterin, 2019). Ces dernières

sont sources de spéculation depuis quelques années, le cours du Bitcoin étant à 327 USD en novembre 2015, 17 000 USD fin décembre 2017 et à 8 600 USD au mois de novembre 2019. Le principe général de ces cryptomonnaies est basé sur le principe de « blockchain » . Ce dernier peut se voir comme un gigantesque livre de comptes publics, distribué et accessible à tous. Afin de rajouter des transactions, il faut « miner » des blocs. Le fait de miner des blocs permet de gagner des éléments de la monnaie. Ainsi, lorsqu'un mineur crée un bloc et que celui-ci est accepté par le système, il gagne des Bitcoins ou du gaz (sous division de l'Etherum). Afin de miner un bloc et donc de gagner de la monnaie, il faut effectuer une preuve de travail (« Proof of Work » ou POW). Dans le cas du Bitcoin, il faut trouver un petit nombre à ajouter au bloc, afin que le hash du bloc commence par un certain nombre de 0. Pour l'Ethereum, il faut effectuer un « contrat intelligent » , où le mineur va devoir exécuter un certain algorithme qui sera défini par le contractant.

Ainsi, l'on voit que pour obtenir ces monnaies, il faut de la puissance de calcul. Plus l'on en possède et plus nous avons de chances d'obtenir des Bitcoins ou de l'Ether. De plus, ces systèmes sont parfaitement conçus pour des systèmes distribués. Or un réseau de zombies est par définition un système distribué où les zombies travaillent en synergie pour accomplir une tâche précise. Ainsi, l'on comprend pourquoi le minage de cryptomonnaies apparaît comme des objectifs d'utilisation d'un réseau de zombies. Plus un réseau possédera de zombies et plus il sera rentable.

2.3 LES OUTILS D'ÉTUDE

Maintenant que nous avons défini les notions importantes et détaillé les objectifs principaux des réseaux de zombies, nous allons expliciter les méthodes et outils utilisés pour les étudier.

2.3.1 LES POTS DE MIEL

Les pots de miel (ou « honeypots ») sont des outils clés pour détecter et analyser les programmes malveillants (Provos *et al.*, 2004; Koroniotis *et al.*, 2019). Ce sont des objets (réel ou virtuels) qui simulent des objets faibles (ordinateur avec une faille, objet connecté sans mot de passe etc.) afin qu'ils puissent se faire infecter par un programme malveillant. Ils vont ensuite garder une copie du code pour analyse future et vont enregistrer toutes les actions que fera le programme malveillant. On distingue principalement deux catégories de pots de miels : ceux à forte interaction et ceux à faible interaction.

La première catégorie correspond à des objets simulant la totalité d'un système vulnérable et pouvant être compromis. La seconde catégorie ne simule que les services souvent attaqués et ne va pas exécuter le logiciel malveillant. L'ensemble de ces techniques permettent de facilement créer des règles de détection permettant d'identifier les futures attaques. Ces outils récoltent énormément de données sur les réseaux de zombies, au niveau de leur comportement et de leurs communications réseau. L'inconvénient majeur est le besoin de stockage qui peut rapidement exploser. De plus, il est extrêmement compliqué de simuler avec exactitude une cible adéquate. C'est encore plus vrai pour les objets connectés, qui peuvent utiliser près d'une dizaine d'architectures différentes pour les processeurs (Pa *et al.*, 2015).

Dans le monde très hétérogène de l'internet des objets, les créateurs de réseaux de zombies créent des versions différentes de leurs logiciels malveillants en fonction des architectures ciblées. On retrouve ainsi des binaires compilés pour les architectures ARM, MIPS, MIPSEL, etc. Cela oblige les chercheurs à adapter leurs pots de miels afin de capter un maximum de binaires différents et ainsi étudier au mieux les menaces. C'est ce qu'on mis en place l'équipe de Pa *et al.* (2015). Ils ont créé un pot de miel mixte, avec une partie présentant une faible interaction afin de détecter et d'enregistrer les attaques sur le protocole Telnet.

Ils ont aussi développé une partie avec une interaction forte, capable de faire tourner les programmes malveillants sur 11 architectures de processeurs différentes. Pour ce faire, ils ont utilisé de nombreuses machines virtuelles afin de simuler au mieux les objets connectés. À l'époque, cela leur a permis de découvrir 39 nouveaux échantillons, inconnus de la base de données VirusTotal. Un autre exemple est le projet SIPHON (Guarnizo *et al.*, 2017), qui met en place plus de 80 objets connectés virtuels à travers le monde. Chacun de ces objets est un pot de miel de forte interaction et utilise 7 objets réels. Leur projet n'a pas été identifié comme un pot de miel par le site Shodan, [2] montrant le réalisme de leur outil.

2.3.2 LES ANALYSE RÉSEAUX

En plus d'analyser le comportement des programmes malveillants en créant des pots de miels, les équipes de recherches analysent l'ensemble des traces réseau possibles afin de détecter les attaques mises en place par les réseaux de zombies. Pour ce faire, ils utilisent principalement trois outils : les télescopes et les scans réseau. L'ensemble de ces outils se base principalement sur des données publiques nécessaires au bon fonctionnement du réseau internet : les requêtes DNS et IP ainsi que le trafic réseau non chiffré. L'ensemble de ces outils ne peut pas analyser du trafic chiffré.

Les télescopes réseaux

Le but principal des télescopes réseau est de surveiller un grand nombre d'adresses IP (Moore *et al.*, 2004). Le télescope va enregistrer l'ensemble du trafic à destination de ces adresses. Lorsque l'on surveille une partie de l'espace IP que l'on sait non assignées, on peut facilement détecter les scans aléatoires produits par les réseaux de zombies. Cela permet aux chercheurs d'estimer en partie l'activité et la taille du réseau de zombies. De plus, en surveillant le trafic

2. `https://shodan.io`

entrant et sortant d'un grand nombre d'adresses IP assigné, il y a de fortes chances pour arriver à détecter les attaques de déni de services, la transmission de logiciel malveillant ou plus rarement les ordres d'attaques.

Ces télescopes enregistrent de très grandes quantités de données et vont rarement pouvoir traiter l'information en temps réel. Par exemple, lors de l'étude sur le programme malveillant Mirai, les équipes de Google et Akamai ont mis en place un télescope réseau enregistrant en moyenne 1,1 million de paquets par minutes, en provenance de 269 000 adresses différentes (Antonakakis *et al.*, 2017).

Ces techniques permettent en général de retracer les évènements lors d'une attaque de grande ampleur. Ce sont des outils d'enquête permettant parfois de remonter la trace des attaquants.

Les scans

La seconde méthode d'analyse réseaux couramment utilisée est le scan actif de l'espace IP. Il existe plusieurs méthodes, par exemple celle développée par Durumeric *et al.* (2015). Cela permet de rapidement récupérer des informations sur les objets vulnérables, notamment leur nature et leur fabricant (Antonakakis *et al.*, 2017). Parfois cela peut aussi donner des informations sur les systèmes d'exploitation utilisés.

Pour conclure, nous observons que les deux grandes catégories d'outils sont complémentaires. En effet, les outils d'analyse réseau permettent d'obtenir des informations sur l'épidémie que représente le réseau de zombies. Grâce à ces outils, les chercheurs peuvent obtenir des données relatives à la taille du réseau et à la nature des objets qui le composent. Les pots de miels, quant à eux, servent surtout à analyser le programme malveillant, ses caractéristiques et son comportement. Si l'on devait faire une analogie avec la médecine, les pots de miels sont les microscopes et les équipements d'analyse des agents pathogènes, là où les outils d'analyse

réseau correspondent aux outils de mesure et de gestion des pandémies.

2.4 PROBLÉMATIQUE ET QUESTION DE RECHERCHE

Nous avons vu tout au long de ces deux chapitres que le monde de l'internet des objets est complexe et hétérogène. Nous nous sommes d'abord intéressés aux objets les plus petits, ayant le moins de capacités de calcul, mais ayant un impact physique sur le monde. Nous avons vu qu'attaquer ces capteurs et effecteurs pose plusieurs difficultés, notamment le besoin de proximité. De plus, il est difficile de créer des attaques de masse en utilisant uniquement ces objets comme vecteurs de transmissions. Nous avons aussi observé qu'il est beaucoup plus simple et rentable, de créer de larges réseaux de zombies en attaquant des objets de plus haut niveau dans notre schéma architectural. En effet, une attaque sur les concentrateurs ou sur les interfaces de contrôle permet de corrompre l'intégralité d'un réseau d'objets connectés. De plus, ces interfaces étant très souvent connectées au réseau Internet, il est possible de facilement propager les logiciels malveillants afin de créer des réseaux de zombies.

Concernant les réseaux en eux mêmes, nous avons mis en évidence, au travers des outils utilisés pour les analyser, qu'ils évoluent. En effet, nous avons constaté une augmentation du nombre d'architectures de processeurs ciblées au cours des dernières années. Cela a obligé les chercheurs à faire évoluer leurs outils et leurs méthodes d'analyse. De ce fait, nous nous sommes concentrés sur l'évolution des réseaux de zombies dans le domaine des objets connectés. En effet, étant donné l'évolution constante des logiciels malveillants ciblant les objets connectés, nous avons souhaité améliorer les méthodes et techniques permettant d'analyser ce phénomène d'évolution. Notre but est de créer un outil permettant de mieux visualiser et comprendre les évolutions de tels programmes, afin de pouvoir mieux les prédire. Ainsi, la question de recherche de ce mémoire est la suivante : quelles sont les évolutions fonctionnelles des logiciels malveillants ciblant l'internet des objets et ayant pour objectif de créer un réseau de zombies ? Nous souhaitons

connaître les évolutions passées de ces programme afin de potentiellement prédire les futures évolutions.

Pour ce faire, nous avons amélioré les taxonomies déjà existantes afin de mieux décrire ces programmes, leurs objectifs et leurs méthodes d'attaques. Puis nous les avons utilisé afin de créer une nouvelle représentation de l'évolution des logiciels malveillants. Enfin, nous avons développé un outil de simulation de propagation des infections de réseaux de zombies. Cela nous permet visualiser les effets que peuvent avoir diverses fonctionnalités et comportements de programmes malveillants sur leurs vitesses de propagation et donc sur leur efficacité. Cet outil nous aide *in fine* à mieux comprendre les impacts de chaque fonctionnalité et ainsi à mieux prédire leurs évolutions.

CHAPITRE 3

LES ÉVOLUTIONS DES LOGICIELS MALVEILLANTS ET DE LEURS RÉSEAUX DE ZOMBIES

Dans ce chapitre, nous allons détailler la méthodologie employée pour créer notre taxonomie. Dans le prochain, nous expliquerons comment nous avons créé notre outil de simulation. Dans un premier temps, nous effectuerons une revue de littérature concernant les taxonomies existantes. Dans une seconde partie, nous détaillerons notre taxonomie ainsi que notre outil de représentation de l'évolution des programmes malveillants. L'ensemble de ces travaux a fait l'objet d'une première publication préliminaire dans le workshop Trsustworthy Computing de la conférence QRS 2019 à Sofia (Benjamin *et al.*, 2019). La totalité des résultats, des codes et algorithmes sont disponibles sur le Github de l'auteur. [1]

3.1 MÉTHODOLOGIE

Dans cette section, nous allons présenter une revue de littérature en rapport avec les taxonomies existantes concernant les programmes malveillants et permettant de décrire leurs caractéristiques ainsi que leurs évolutions.

1. `https://github.com/bvignau/The_Botnet_Game`

3.1.1 LES TAXONOMIES DES RÉSEAUX DE ZOMBIES

Afin de représenter au mieux l'évolution des programmes malveillants ciblant l'internet des objets et les réseaux de zombies associés, nous avons analysé deux choses. La première est l'ensemble des taxonomies existantes sur le sujet. La seconde est l'ensemble des fonctionnalités développées par ces programmes malveillants. En effet, notre but est de pouvoir identifier l'ensemble des fonctionnalités qui représentent ces programmes, montrer leurs changements, leurs modifications et essayer de les prédire. Ainsi, notre taxonomie se doit de comporter plusieurs niveaux d'abstraction et doit être extensible afin de s'adapter aux nouvelles fonctionnalités qui seront développées dans le futur. Nous avons étudié les taxonomies de De Donno *et al.* (2018b), Hachem *et al.* (2011), Dagon *et al.* (2007) et Rawat *et al.* (2018).

Pour les analyses, nous nous sommes basés sur divers critères. Le premier, correspond au nombre de niveaux d'abstraction. Nous souhaitons que l'identification d'un niveau permette bien d'identifier l'ensemble des taxons fils. Le second critère correspondra au nombre de fonctionnalités ou de comportements qui sera décrit. Le troisième sera le nombre de fonctionnalités pertinentes spécifiques aux objets connectés.

Afin de compléter notre taxonomie, nous avons étudié un maximum de programmes malveillants ciblant les objets connectés afin de créer des réseaux de zombies. Ainsi, nous avons effectué des recherches générales sur ces logiciels afin de les lister. Pour ce faire, nous avons effectué des recherches sur les moteurs classiques tels que Google Scholar ou Web of Science. Nous avons aussi étudié les sources citées par les articles précédemment sélectionnés. Enfin, nous avons effectué des recherches précises pour chaque famille de programmes malveillants, en étudiant les sources académiques, les rapports techniques des entreprises ou instituts ayant découvert ou analysé certains de ces programmes. Enfin, nous avons analysé les codes source lorsque ces derniers étaient disponibles. Au total, nous avons obtenu suffisamment de données

exploitables pour analyser 16 familles de programmes malveillants. Cela nous a permis d'extraire 46 fonctionnalités différentes.

3.1.2 LES OUTILS DE MODÉLISATION DE LA PROPAGATION DES RÉSEAUX DE ZOMBIES

Après avoir mis en place notre taxonomie, nous avons souhaité modéliser l'impact de certaines fonctionnalités sur le fonctionnement et l'efficacité des réseaux de zombies. Pour cette partie, nous nous sommes concentrés sur la vitesse de propagation des réseaux de zombies ainsi que sur leurs capacités à infecter un maximum de victimes. Nous avons par la suite souhaité modéliser le phénomène de concurrence entre les divers réseaux de zombies. En effet, chaque réseau souhaite infecter un maximum d'hôtes, or ce nombre est limité. De plus, il est très fréquent que certains objets soient vulnérables à plusieurs attaques ou que plusieurs réseaux de zombies utilisent le même vecteur d'attaque pour se propager. Par exemple, l'attaque par force brute des identifiants de connexion sur les services SSH et Telnet sont très utilisés par différents réseaux de zombies. Cela mène donc à une concurrence entre les réseaux de zombies, qui se doivent de monopoliser un maximum de ressources afin de maximiser leur puissance d'attaque et leur rentabilité.

Avant de créer notre logiciel, nous avons effectué une petite analyse des modèles existants. Notre objectif étant de modéliser le phénomène de propagation des programmes, nous avons donc étudié les modèles infectieux existants (Wainwright et Kettani, 2019; Zou *et al.*, 2002; Chen *et al.*, 2003; Chen et Ji, 2005; Abaid *et al.*, 2016; Ji *et al.*, 2018). Ces modèles sont basés sur ceux utilisés en médecine (Daley et Gani, 1999) afin d'analyser et prédire les phénomènes d'épidémie.

3.2 ÉTUDES DES TAXONOMIES EXISTANTES ET DES FAMILLES DE PROGRAMMES MALVEILLANTS

Nous allons détailler dans cette section la construction de notre nouvelle taxonomie ainsi que son utilisation afin de représenter l'évolution des programmes malveillants. Tout d'abord, nous allons analyser les taxonomies existantes et nous allons ensuite construire la nôtre. Enfin, nous exposerons nos représentations graphiques.

3.2.1 LES TAXONOMIES EXISTANTES

Les taxonomies étudiées ici ne sont pas forcément spécialisées pour les réseaux de zombies constitués d'objets connectés, mais sont tout de même très utiles. Nous allons ici les analyser par ordre chronologique de publication. Nous avons étudié les taxonomies de réseaux de zombies les plus abouties et les plus utilisées.

La taxonomie de Dagon *et al.* (2007) se concentre sur les structures de réseaux de zombie, en fonction de leur utilité pour l'attaquant. Cela permet, selon eux, de pouvoir définir des réponses adaptées à chaque réseau. Leur taxonomie se base sur quatre critères : l'efficacité des attaques, la bande passante disponible, l'efficacité des communications ainsi que sur la robustesse du réseau. Pour ce faire, ils mettent en place quatre métriques pour mesurer ces critères. On a donc respectivement, la taille du réseau se comptant en nombre de victimes, la bande passante moyenne disponible à n'importe quel moment, le plus long chemin utilisé pour transmettre un message a tous les zombies et enfin les mécanismes de redondance.

Dans leur article les chercheurs analysent différentes formes de réseaux possibles comme les réseaux aléatoires d'Erdös-Rényi, les réseaux pair-à-pair ou les modèles de Watts-Strogatz. Cependant, leur taxonomie ne se base pas sur des réseaux existants et ne prend pas en compte

les architectures centralisées. Ils utilisent principalement des réseaux de zombies expérimentaux afin de détailler les meilleures fonctionnalités.

Cette taxonomie est ancienne et ne représente que peu de fonctionnalités réellement observées. Cependant, l'idée de déterminer l'efficacité d'un réseau de zombies par son nombre de victimes est intéressante. Bien que nous ne pouvons pas l'utiliser pour décrire le comportement d'un réseau de zombie, nous pouvons nous en servir pour justifier en partie, la persistance ou la disparition d'un comportement. En effet, si une fonctionnalité ou un comportement donné permet d'augmenter l'efficacité générale du réseau de zombies, alors ce dernier aura sûrement plus de chances de rester et se retrouvera probablement dans plusieurs réseaux de zombies. Cette métrique de l'efficacité peut aussi nous renseigner sur la dangerosité d'un réseau. En effet, plus ce dernier contient de zombies, plus il pourra monopoliser de la bande passante et plus ses attaques seront puissantes.

La seconde taxonomie que nous avons étudiée est celle de Hachem *et al.* (2011). Celle-ci se base sur le cycle de vie des réseaux de zombies afin de déterminer ses caractéristiques. Ainsi, les auteurs découpent leur taxonomie en deux niveaux. Le premier contient quatre éléments, directement dérivé du cycle de vie. Le second niveau correspond aux fonctionnalités observées chez certains programmes malveillants et leurs réseaux de zombie associés. L'élément du premier niveau est : la propagation et l'infection, le système de commande et de contrôle, l'application et enfin la résilience du réseau.

La première partie contient des fonctionnalités comme l'utilisation de mails frauduleux, l'exploitation de failles logicielles, l'envoi de liens corrompus par messagerie instantanée, l'utilisation de fichiers corrompus dans des réseaux de partages P2P ou encore l'utilisation d'autres réseaux de zombies. Cette dernière fonctionnalité regroupe la mise à jour d'un réseau ainsi que son emplacement afin de propager un ou plusieurs programmes malveillants différents. Ici, on peut noter que seules les fonctionnalités d'exploitation de failles et d'utilisation d'autres réseaux de

zombies peuvent être utiles pour créer un réseau d'objets connectés.

Concernant le système de commande et de contrôle, l'équipe détaille deux modèles principaux : la forme centralisée et la forme décentralisée. La première forme étant plus efficace, mais possédant un point de défaillance unique, contrairement à la seconde. Ensuite, ils rajoutent la topologie des réseaux et les protocoles de communications internes. Ici, l'ensemble des caractéristiques décrites peuvent être retrouvées dans un réseau de zombies d'objets connectés.

La partie application concerne les objectifs et les attaques possibles du réseau de zombies. Ce sont ces fonctionnalités qui provoquent le plus de dommages à nos sociétés. On y retrouve principalement les attaques de déni de services distribuées, la distribution de spam, l'espionnage et l'hébergement d'activités malicieuses.

La dernière partie correspond à toutes les fonctionnalités annexes du réseau de zombies. Ces dernières vont lui permettre d'être plus efficace, de se faire repérer moins facilement ou d'être plus résilient. On retrouve ainsi des fonctionnalités de mise à jour, d'obfuscation, etc.

L'ensemble de cette taxonomie est très efficace : on a une représentation où chaque noeud représente correctement l'ensemble de ses fils. Cette taxonomie est basée sur des données empiriques et peut facilement être étendue. Cependant, de par son ancienneté et sa généralité, elle ne peut décrire la totalité des réseaux de zombies d'objets connectés. Enfin, l'utilisation du cycle de vie du réseau pour créer les catégories de fonctionnalités est extrêmement pertinente. Cela permet de créer des réponses adaptées pour chaque étape de la vie du réseau de zombies.

Cette taxonomie comporte donc deux niveaux de hiérarchiques, 42 taxons qui ont tous été identifiés dans des réseaux de zombies. Cette dernière a été appliquée pour décrire 16 familles de réseaux de zombies, apparues entre 1999 et 2009. Cependant, aucun de ces réseaux de zombies n'était composé d'objets connectés, on ne retrouve donc aucune fonctionnalité spécifique à ces objets.

La taxonomie de De Donno *et al.* (2018b) se concentre sur les réseaux de zombies composés d'objets connectés et effectuant des attaques de déni de services. Cette dernière est composée de 4 niveaux hiérarchiques et utilise 44 taxons. Contrairement à la taxonomie précédemment étudiée, le premier niveau ne correspond pas aux différentes phases de vie des réseaux de zombie. Ce sont directement les familles de fonctionnalités qui y sont décrites. On y retrouve ainsi des familles de fonctionnalités en rapport direct avec les attaques de déni de services comme le taux d'attaque, les adresses IP sources utilisées durant l'attaque, les ressources utilisées etc. On trouve aussi le regroupement des architectures utilisées, les méthodes de recherche de victimes ou de propagation. Seules deux familles utilisent quatre niveaux hiérarchiques ; la majorité n'en utilise que deux.

Cette taxonomie est très intéressante, car elle permet de mettre en avant de nouvelles fonctionnalités non décrites par les précédentes. C'est le cas notamment des fonctionnalités de recherche de victimes. Les chercheurs montrent que plusieurs techniques sont employées. Ainsi, pour déterminer les adresses à scanner, on retrouve l'utilisation de listes prédéfinies, la génération aléatoire ou l'utilisation de permutations.

Ici aussi, la taxonomie se base sur des fonctionnalités observées dans 13 familles de réseaux de zombies composés d'objets connectés. Cependant, l'ensemble des fonctionnalités décrites ne sont pas spécifiques à ces réseaux et se retrouvent dans les autres réseaux, plus anciens ou composés d'autres éléments. De plus, cette taxonomie est particulièrement précise dans sa description des attaques de déni de services. En effet, cette dernière décrit les victimes des attaques de déni de services, la distribution du trafic de l'attaque ou l'évolution du taux des attaques. Or ce genre de taxonomie se prête plus à une description poussée de tout type d'attaque de déni de services et ne permet pas de décrire chaque type d'attaque comme une fonctionnalité propre du réseau de zombies.

Enfin, les chercheurs ont utilisé cette taxonomie pour mettre en évidence des corrélations entre

les réseaux de zombies. Cela permet de montrer de premiers liens d'influence et d'évolution. Cependant, ces derniers ne sont pas précis et l'on ne sait pas en quoi chaque programme influence les autres.

La dernière taxonomie que nous avons analysée est celle de Rawat *et al.* (2018). Cette dernière porte principalement sur les réseaux de zombies ayant une architecture décentralisée. Ils présentent ainsi une courte taxonomie des architectures ainsi qu'une taxonomie des méthodes utilisées pour détecter ce type de réseaux. Ici, nous ne considérons que la taxonomie en rapport avec l'architecture des réseaux de zombies. Celle-ci comporte deux niveaux. Le premier est le type d'architecture, comprenant les réseaux pair-à-pair (P2P) structurés, non structurés, hybrides et avec super pairs. Le second niveau n'est utilisé que pour différencier certains protocoles pouvant être utilisés comme fonctionnalités du premier niveau. Au total, cette taxonomie comporte quatre taxons.

L'ensemble des diverses analyses est résumé au tableau 3.1

Taxonomie	Nombre de niveaux	Nombre de taxons	Taxons spécifiques IoT
De Donno *et al.* (2018b)	4	44	0
Hachem *et al.* (2011)	2	42	0
Dagon *et al.* (2007)	2	7	0
Rawat *et al.* (2018)	2	5	0
Notre modèle	3	46	8

Tableau 3.1 – Comparaison des différentes taxonomies

En conclusion, les taxonomies existantes permettent de fournir une base intéressante pour la création d'une taxonomie des réseaux de zombies IoT. Cependant, on peut observer que deux de ces taxonomies n'ont pas suffisamment de taxons pour correctement classifier notre population

de réseaux de zombies. Les deux autres ayant plus de 40 taxons différents ne possède aucun taxons spécifiques aux réseaux de zombies d'objets connectés. C'est pour cette raison, que nous avons repris en parties ces taxonomies, et y avons rajouter des taxons spécifiques à ces réseaux de zombies.

Notre taxonomie est donc plus efficace pour décrire l'ensemble des fonctionnalités observées sur les réseaux de zombies d'objets connectés que nous avons étudié. Cependant, cette dernière devra être améliorée au fur et à mesure que nous découvrirons d'autres réseaux de zombies. En effet, il n'est pas impossible que ces derniers innovent et ajoutent de nouveaux comportements au fil des années. À ce moment-là, il faudra rajouter des taxons permettant de décrire ces nouveaux comportements. L'avantage de notre taxonomie est son extensibilité. Du fait qu'elle est basée sur le cycle de vie des réseaux de zombies, il sera toujours possible d'y rajouter des taxons décrivant de nouveaux comportements. Une autre méthode pour compléter cette taxonomie serait d'imaginer de nouveaux comportements, de les simuler afin de déterminer s'ils sont utilisables par un réseau de zombies, puis d'ajouter le comportement à notre taxonomie. Afin de décrire l'évolution de ces réseaux, il nous faut un maximum de taxons permettant de décrire au mieux chaque réseau et ainsi pouvoir identifier facilement les points communs et différences de chacun.

3.2.2 ORGANISER LES DIVERS COMPORTEMENTS

Afin de créer notre taxonomie, nous avons repris certaines idées des articles précédemment analysés. Nous en avons aussi filtré une partie, puis nous avons étudié 16 familles de réseaux de zombies d'objets connectés. Notre but étant de construire une taxonomie permettant de décrire efficacement les fonctionnalités et les comportements de chaque programme malveillant et de son réseau de zombies associé, une taxonomie hiérarchisée en plusieurs niveaux nous a paru pertinente. Afin que chaque partie puisse aider à la compréhension de ces réseaux et permette

de trouver des réponses adéquates, nous avons repris l'idée de Hachem *et al.* (2011) et nous utilisons aussi le cycle de vie pour définir notre premier niveau.

Nous souhaitons que chaque niveau représente une abstraction des niveaux inférieurs. Ainsi, réussir à représenter complètement un niveau, par un algorithme, un automate fini ou tout autre objet mathématique, reviendrait à identifier l'ensemble des taxons de ses niveaux inférieurs.

Enfin, notre taxonomie doit présenter des taxons spécifiques aux réseaux de zombies d'objets connectés ainsi que des taxons communs à l'ensemble des réseaux de zombies. Ici, nous nous limitons aux fonctionnalités observées dans les réseaux de zombies d'objets connectés que nous avons étudiées. Par exemple, nous n'inclurons pas la méthode de propagation par messagerie instantanée, car celle-ci n'a jamais été utilisée dans les familles de programmes malveillants que nous avons étudiées. Cependant, notre taxonomie doit pouvoir s'étendre pour ajouter de telles fonctionnalités et ainsi décrire l'ensemble des réseaux de zombies existant. De la même manière, notre taxonomie doit être extensible pour supporter l'apparition de nouvelles fonctionnalités, observées dans de futurs réseaux malicieux ou prévues par des équipes de recherches à des fins de protections.

3.2.3 ANALYSE DES FAMILLES DE PROGRAMMES MALVEILLANTS

Les familles analysées sont celles où nous avons pu trouver le plus de données, que ce soit dans la littérature académique ou technique. En effet, il faut noter que certaines familles de réseaux de zombies ont reçu plus d'intérêt que d'autres. Ce fut notamment le cas du programme Mirai, qui de par la puissance de ces attaques et la libération de son code source a entraîné la création de nombreux réplicats. Nous allons ici décrire brièvement chaque programme malveillant retenu pour l'étude. L'ensemble des sources associées à chaque programme est donné dans le tableau 3.2.

Linux.Hydra (2008)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Janus, 2011), (De Donno *et al.*, 2018b) (Botticelli, 2017), (Angrishi, 2017), (Zaddach, 2018), (De Donno *et al.*, 2018b), ifding (2017).

Linux.Hydra est, à la base, un outil libre de droit fait pour créer des réseaux de routeurs-zombies. Il permettait de créer un programme malveillant pour propager l'infection. Cet outil permettait aussi de contrôler le réseau de zombies via un chat IRC et de l'utiliser pour lancer des attaques de déni de service distribuées. Cet outil est apparu en 2008 sur les réseaux cachés de l'Internet (dark web). Il possède les fonctionnalités vitales pour un réseau de zombies, mais ces dernières ne sont pas très développées.

Psyb0t (2009)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Ďurfina *et al.*, 2013; DroneBL, 2009; Janus, 2011) De Donno *et al.* (2018b) (Angrishi, 2017; Botticelli, 2017; Celeda *et al.*, 2010; Zaddach, 2018; De Donno *et al.*, 2018b).

Psyb0t est le premier programme malveillant ciblant des routeurs et des modems trouvés par une équipe de chercheurs. Il tire parti des attaques par dictionnaires afin de trouver les identifiants de plusieurs services présents chez les victimes. Il utilise aussi l'exploit présent chez de nombreux produits de D-link, permettant d'obtenir le mot de passe en envoyant une requête particulière au serveur web. Cette vulnérabilité n'a jamais été considérée comme une CVE (Common Vulnerability Exposure). C'est aussi un réseau de zombies utilisant les chats IRC pour communiquer. Son objectif est de créer des attaques de déni de service. Le réseau a été désactivé par l'auteur, qui a envoyé une commande afin de tuer tous les zombies.

Chuck Norris (2009)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Celeda *et al.*, 2010; Janus, 2011; De Donno *et al.*, 2018b) (Celeda *et al.*, 2010; Botticelli, 2017; Zaddach, 2018; De Donno *et al.*, 2018b).

Quelques mois après la fin de Psyb0t, un nouveau réseau de zombies a fait son apparition. Il fut appelé « Chuck Norris », car le code binaire contenait la chaine de caractère : « In nome di Chuck Norris » (Au nom de Chuck Norris).Ce programme malveillant possède beaucoup de points communs avec Psyb0t, par exemple la manière d'encoder les informations sensibles, les fonctionnalités de communication, de scan et d'infection. Les chercheurs pensent que les auteurs de Chuck Norris sont probablement les mêmes que ceux ayant créé Psyb0t.

Tsunami (2010)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Janus, 2011; De Donno *et al.*, 2018b) (Botticelli, 2017; De Donno *et al.*, 2018b) (unlnow, 2015).

Les chercheurs pensent que le programme Tsunami, apparu un an après Chuck Norris, est une évolution de ce dernier. En effet, les méthodes d'obfuscation sont encore les mêmes et certaines chaines de caractères et adresses IP sont identiques. Cette nouvelle version a été nommée Tsunami/ Kaiten en raison de son utilisation du protocole libre de déni de service distribué du même nom.

Linux.Aidra (2012)

Pour étudier ce programme nous avons utilisé les sources suivantes : (De Donno *et al.*, 2018b; Botticelli, 2017) (Botticelli, 2017) (ifding, 2017).

Linux.Aidra, aussi connu comme Linux.LightAidra a été découvert en 2012 par une équipe de chercheurs. Ils ont observé un grand nombre d'attaques « Telnet » (tentative d'accès au service telenet via une attaque par dictionnaire) en provenance de routeurs, télévisions connectées, caméras connectées, etc. Il fut le premier à attaquer les architectures ARM et à utiliser le « cross compilation » (compilation d'un programme en un binaire destiné à une autre architecture que le processeur de la machine compilant le programme) afin de pouvoir infecter plus de victimes.

Carna (2012)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Carna, 2012) (Angrishi, 2017; Botticelli, 2017).

Le réseau de zombies Carna est une sorte d'étude scientifique pour laquelle les auteurs ont scanné la totalité du réseau Internet durant 24h. Ils ont utilisé ce réseau pour faire une carte détaillée des connexions Internet dans le monde. Tout comme Aidra il possède des techniques de « cross compilation » et peut cibler les architectures ARM. C'est le premier programme malveillant ciblant les objets connectés mettant en place un réseau de zombies qui ne lance pas d'attaques de déni de service.

Linux.Darlloz (2014)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Botticelli, 2017; Manoharan, 2018; Hayashi, 2013) (Angrishi, 2017; Botticelli, 2017; Manoharan, 2018; Hayashi, 2014).

Linux.Darlloz est un ver ciblant principalement les systèmes Linux et infectant routeurs et caméras connectées. Il est le premier à utiliser un exploit en provenance d'une CVE pour infiltrer ses victimes. En 2014 des chercheurs ont remarqué que le réseau de zombies avait commencé à

miner des crypto monnaies telles que le Mincoin et le Dogecoin.

Linux.Wifatch (2014)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Ballano, 2015) (Ballano, 2015; Zaddach, 2018) (team, 2016).

Linux.Wifatch est un réseau de zombies créé par une équipe de « white hat ». Leur but est de faire de la prévention et de sécuriser les objets utilisés par d'autres programmes malveillants. Ainsi, Wifatch utilise un dictionnaire de mot de passe pour s'introduire dans des objets connectés, supprimer les programmes malveillants plus anciens et stopper les services tels que Telnet et SSH. Il met ensuite un message dans les logs afin d'inviter le propriétaire de l'objet à changer de mot de passe.

Bashlite (2014)

Pour étudier ce programme nous avons utilisé les sources suivantes : (De Donno *et al.*, 2018b; Marzano *et al.*, 2018) (Zaddach, 2018; Botticelli, 2017; Manoharan, 2018; Angrishi, 2017) (ifding, 2017).

Bashlite, aussi appelé Qbot, Gayfgt, Lizkebab ou Torlus est l'un des programmes malveillants les plus connus dans le monde des objets connectés. Il a été très utilisé par plusieurs entités malveillantes afin de créer des réseaux de zombies très puissants. La première version a été découverte en 2014 et son code source a été libéré en 2015. Certaines variantes ont réussi à infecter plus de 100 000 objets connectés. Ce programme a servi de base pour créer Mirai.

Remaiten (2016)

Pour étudier ce programme nous avons utilisé les sources suivantes : (De Donno *et al.*, 2018b; Manoharan, 2018; Malik et M.Léveillé, 2016) (Zaddach, 2018; Botticelli, 2017; Manoharan, 2018; Angrishi, 2017).

Remaiten est en quelque sorte une fusion entre Tsunami et Bashlite. Il utilise plusieurs fonctionnalités de ces derniers, comme le protocole d'attaque DDoS de Tsunami ou le scanner de Bashlite. Cependant, il ajoute une nouvelle fonctionnalité, capable de détecter l'architecture de la victime et de ne lui envoyer que le bon binaire. Avant, les réseaux de zombies, envoyaient tous les binaires qu'ils possédaient et les essayaient tous jusqu'à réussite.

Mirai (2016)

Pour étudier ce programme nous avons utilisé les sources suivantes : (De Donno *et al.*, 2018b; Kolias *et al.*, 2017; Ji *et al.*, 2018; Antonakakis *et al.*, 2017) (Kolias *et al.*, 2017; De Donno *et al.*, 2018b; Ji *et al.*, 2018; Manoharan, 2018; Botticelli, 2017; Marzano *et al.*, 2018; Angrishi, 2017; Zaddach, 2018; Antonakakis *et al.*, 2017) ifding (2017).

Mirai est sûrement le plus connu et le plus étudié des programmes malveillants ciblant les objets connectés. Il a été créé pour surpasser Bashlite et a produit les plus grosses attaques de déni de service jamais enregistrées. Il a attaqué le blog de l'expert en sécurité informatique Krebs avec une attaque à 650 Gb/s, des serveurs Minecraft chez OVH avec une puissance de 1.2Tbps . Le service de nom de domaine DYN a aussi subi une grosse attaque, rendant indisponible des services tels que Facebook ou Netflix durant plusieurs heures. Son code source a été dévoilé peu de temps après ces attaques et de nombreuses variantes sont apparues et sont encore actives aujourd'hui. Le nom Mirai signifie futur en japonais, l'auteur a donné ce nom en référence à la

série d'animation japonaise « Mirai Nikki ».

Hajime (2016)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Sam Edwards, 2016; Manoharan, 2018) (Manoharan, 2018; Botticelli, 2017; Zaddach, 2018; Sam Edwards, 2016).

Hajime est un réseau de zombies décentralisé apparu à peu près en même temps que Mirai. Son objectif est d'infecter un maximum d'objets connectés. Il va ensuite fermer les ports d'accès aux services, comme le faisait Wifatch. Les chercheurs ne sont pas certains de son objectif final. Certains pensent qu'il peut servir à propager d'autres logiciels malveillants. D'autres pensent qu'il a pour objectif de protéger des objets contre les réseaux Mirai.

Amnesia (2017)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Claud et Cong, 2017) (Claud et Cong, 2017; Leyden, 2017; Manoharan, 2018; Botticelli, 2017).

Le programme malveillant Amnesia utilise une vulnérabilité d'exécution de code distant, présente dans environ 227 000 appareils d'enregistrement vidéo (DVR) fabriqués par TVT Digital. Il est le premier à mettre en place une détection d'environnement virtualisé. Il est ainsi, capable de détecter les « pots de miel » et de supprimer toute la mémoire de la machine virtuelle.

BrickerBot (2017)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Radaware, 2017; Manoharan, 2018; Anubhav, 2017) (Kolias *et al.*, 2017; Cimpanu, 2017; Zaddach, 2018; Geenens, 2017).

BrickerBot est le premier programme malveillant à mettre en place un déni de service physique. En effet, une fois qu'il a infecté une victime, BrickerBot va essayer de détruire l'objet en supprimant la mémoire. Pour le faire, il va écrire des données aléatoires sur l'ensemble des périphériques de stockage de l'objet. Ce programme exploite des CVE afin d'infecter ses victimes.

IoT Reaper (2017)

Pour étudier ce programme nous avons utilisé les sources suivantes : (Point, 2017; System, 2018; yegenshen, 2017) (FortiGuard, 2017; Krebs, 2017).

IoT Reaper (aussi appelé IoTroop) est l'un des plus dangereux programmes malveillants ciblant les objets connectés. Il est capable d'utiliser plusieurs exploits dérivés de plusieurs CVE afin d'infecter un maximum d'hôtes. C'est le premier capable d'utiliser neuf exploits différents.

VPNFilter (2018)

Pour étudier ce programme nous avons utilisé les sources suivantes : (William, 2018a,b; Edmund, 2018) (Cimpanu, 2018a,b).

VPNFilter est le programme malveillant le plus complexe et le plus dangereux que nous avons étudié. Il est considéré par le FBI comme une Menace Avancée Persistante (Advanced Persistance Threat) créé par la Russie. Ce programme est capable de surveiller des contrôleurs industriels SCADA, de mettre en place un VPN à l'intérieur du réseau d'une entreprise et même de détecter des exécutables Windows dans les communications réseau. Ce programme aurait été utilisé en 2018 par la Russie, contre l'Ukraine. De plus, il aurait infecté un grand nombre d'objets connectés.

Famille	Détails techniques	Informations générales	Code Source
Linux.Hydra (1)	(Janus, 2011) (De Donno *et al.*, 2018b)	(Botticelli, 2017) (Zaddach, 2018) (Angrishi, 2017) (De Donno *et al.*, 2018b)	(ifding, 2017)
Psyb0t (2)	(Ďurfina *et al.*, 2013) (DroneBL, 2009) (Janus, 2011) (De Donno *et al.*, 2018b)	(Angrishi, 2017) (Botticelli, 2017) (Celeda *et al.*, 2010) (Zaddach, 2018) (De Donno *et al.*, 2018b)	non disponible
Chuck Norris (3)	(Janus, 2011) (De Donno *et al.*, 2018b) (Celeda *et al.*, 2010)	(Celeda *et al.*, 2010) (Botticelli, 2017) (Zaddach, 2018) (De Donno *et al.*, 2018b)	non disponible
Tsunami/ Kaiten (4)	(Janus, 2011) (De Donno *et al.*, 2018b)	(Botticelli, 2017) (De Donno *et al.*, 2018b)	(unlnow, 2015)
Aidra (5)	(Botticelli, 2017) (De Donno *et al.*, 2018b)	(Botticelli, 2017)	(ifding, 2017)
Carna (6)	(Carna, 2012)	(Angrishi, 2017) (De Donno *et al.*, 2018b)	non disponible
Linux.Darlloz (7)	(Botticelli, 2017) (Hayashi, 2013) (Manoharan, 2018)	(Angrishi, 2017) (Botticelli, 2017) (Manoharan, 2018) (Hayashi, 2014)	non disponible
Linux.wifatch (8)	(Ballano, 2015)	(Ballano, 2015) (Zaddach, 2018)	(team, 2016)

Tableau 3.2 – Sources utilisées pour analyser les familles de réseaux de zombies (1/3)

3.3 NOTRE TAXONOMIE

Dans cette section nous allons expliquer la majorité des taxons. Nous n'allons pas forcément rentrer dans les détails précis de chaque fonctionnalité. Le but ici, est de comprendre les fonctionnalités générales décrites par nos taxons. L'ensemble de ces derniers ainsi que la liste des programmes les implémentant sont donnés dans le tableau 3.5.

Famille	Détails techniques	Informations générales	Code Source
Bashlite (9)	(De Donno *et al.*, 2018b) (Marzano *et al.*, 2018)	(Zaddach, 2018) (Botticelli, 2017) (Angrishi, 2017) (Manoharan, 2018)	(ifding, 2017)
Remaiten (10)	(De Donno *et al.*, 2018b) (Manoharan, 2018) (Malik et M.Léveillé, 2016)	(Manoharan, 2018) (Zaddach, 2018) (Botticelli, 2017) (Angrishi, 2017)	non disponible
Hajime (11)	(Sam Edwards, 2016) (Manoharan, 2018) (Botticelli, 2017)	(Zaddach, 2018) (Manoharan, 2018) (Sam Edwards, 2016)	non disponible
Mirai (12)	(De Donno *et al.*, 2018b) (Kolias *et al.*, 2017) (**?**) (Antonakakis *et al.*, 2017)	(Kolias *et al.*, 2017) (Zaddach, 2018) (Ji *et al.*, 2018) (De Donno *et al.*, 2018b) (Manoharan, 2018) (Marzano *et al.*, 2018) (Angrishi, 2017) (Botticelli, 2017) (Antonakakis *et al.*, 2017)	ifding (2017)

Tableau 3.3 – Sources utilisées pour analyser les familles de réseaux de zombies (2/3)

3.3.1 EXPLICATION DES TAXONS

Pour notre taxonomie, nous nous sommes inspirés de Hachem *et al.* (2011). En effet, notre premier niveau est très similaire au sien et comprend quatre familles correspondant aux trois grandes phases de vie du réseau et une correspondante aux fonctionnalités annexes permettant d'améliorer l'efficacité générale du réseau. Ainsi, nous avons la partie correspondant au recrutement ou à l'infection, la partie de l'organisation et des communications du réseau de zombies et enfin la partie application.

La première partie contient trois sous-parties correspondant à la stratégie de scan, les architectures des victimes et enfin les méthodes d'exploitations. Chaque partie contient des taxons

Famille	Détails techniques	Informations générales	Code Source
Amnesia (13)	(Claud et Cong, 2017)	(Claud et Cong, 2017) (Manoharan, 2018) (Botticelli, 2017) (Leyden, 2017)	non disponible
BrickerBot (14)	(Manoharan, 2018) (Radaware, 2017) (Anubhav, 2017)	(Cimpanu, 2017) (Kolias *et al.*, 2017) (Zaddach, 2018) (Geenens, 2017)	non disponible
IoTReaper (15)	(Point, 2017) (System, 2018) (yegenshen, 2017)	(Krebs, 2017) (FortiGuard, 2017)	non disponible
VPNFilter (18)	(William, 2018b) (William, 2018a) (Edmund, 2018)	(Cimpanu, 2018b) (Cimpanu, 2018a)	non disponible

Tableau 3.4 – Sources utilisées pour analyser les familles de réseaux de zombies (3/3)

représentant des fonctionnalités finales. Notre deuxième partie contient deux sous-parties contenant les trois formes d'architecture de réseaux de zombies que nous avons observés. Il est fort probable que, dans une future extension de ces travaux, cette partie voit apparaître de nouveaux niveaux d'abstraction avec l'apparition d'une diversité dans les architectures de réseaux de zombies.

Notre partie applicative correspond aux fonctionnalités représentant les buts des réseaux de zombies. Ainsi, on distingue quatre grandes familles d'objectifs. La première est le déni de service temporaire, représenté ici par la famille "DDoS". Cette famille contient l'ensemble des fonctionnalités observées représentant les attaques de déni de service distribuées. Nous caractérisons ces attaques de déni temporaires, car la perte de service s'arrête avec l'arrêt de l'attaque, contrairement aux attaques de déni permanents, où le déni de service persiste après l'arrêt de l'attaque. Cette dernière catégorie est représentée par le niveau "PDoS" et ne contient que deux fonctionnalités. La troisième famille correspond à l'espionnage et au vol de données. La dernière famille représente les fonctionnalités de rentabilité du réseau de zombies. Ici, nous avons observé uniquement des fonctionnalités de minage de cryptomonnaies.

Enfin, notre dernière partie contient l'ensemble des fonctionnalités permettant d'améliorer l'efficacité d'un réseau de zombies, par exemple le changement de nom du processus malicieux, le système de mise à jour du programme malveillant ou encore la persistance de ce dernier. Une première visualisation des deux premiers niveaux est donnée à la figure 3.1. Nous allons maintenant expliciter une partie des taxons. Les noms des taxons utilisés dans la figure 3.1 sont donnés au tableau 3.5.

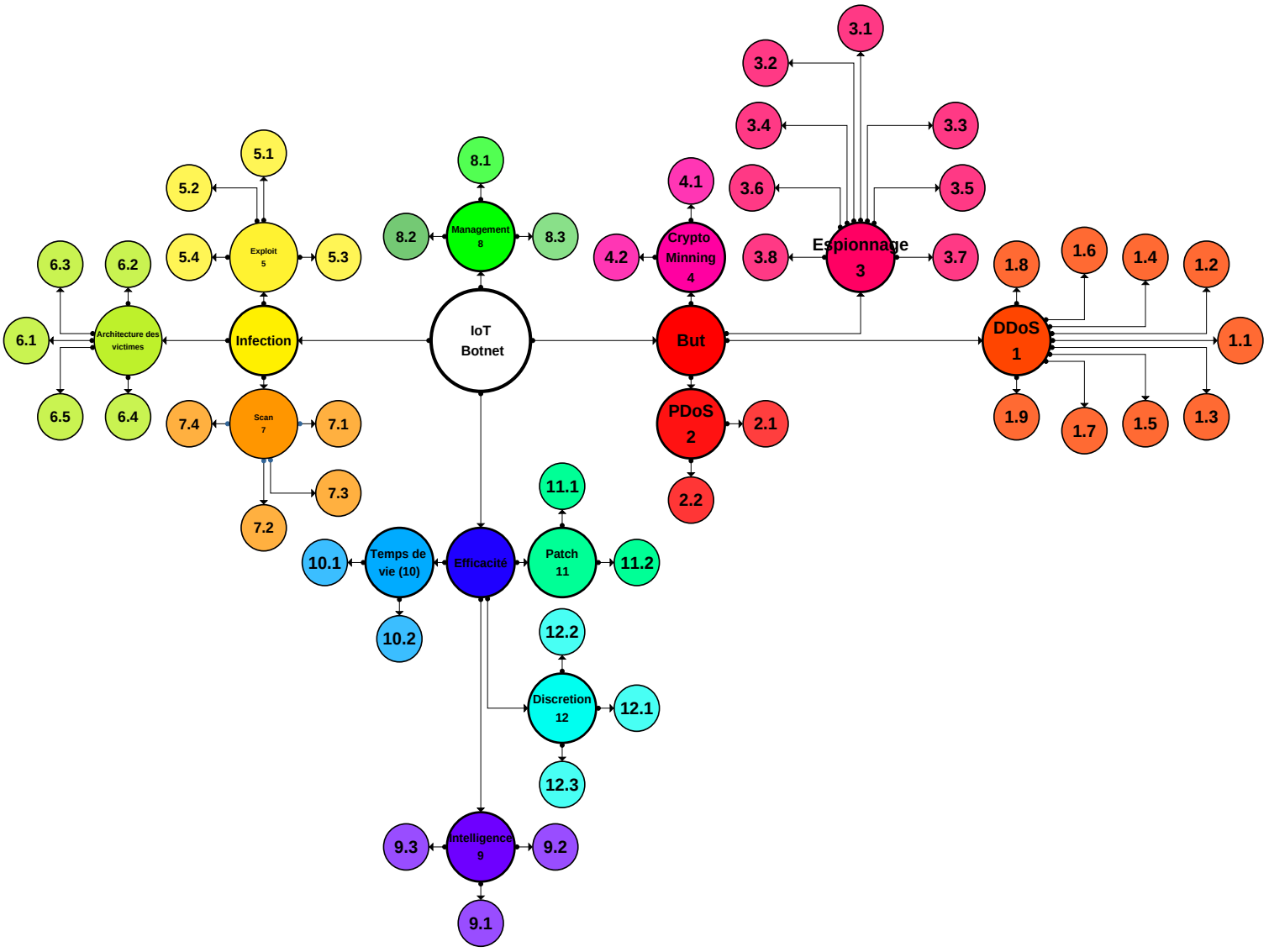

Figure 3.1 – Taxonomie des réseaux de zombies IoT

Famille (1) : déni de service temporaire (DDoS)

La majeure partie des réseaux de zombies de notre étude ont pour objectif de créer des attaques de déni de service. Ainsi, c'est la famille de fonctionnalités la plus diversifiée. On y retrouve les attaques classiques de déni de service tel que le SYN Flood (1.1), l'UDP Flood (1.2), le Ping Flood (1.3), le ACK Flood (1.4), le HTTP Flood (1.6) et le TCP XMAS (1.7). Le but de ces attaques est que chaque zombie envoie un maximum de paquets à une cible particulière afin de saturer sa bande passante. Les fonctionnalités varient avec l'implémentation qui va utiliser divers protocoles comme TCP (1.1, 1.4, 1.5, 1.7) UDP (1.2) ou HTTP (1.6) .

On trouve dans cette famille les attaques de « DNS water torture » ou « DNS waterbording » (1.8). Le but de cette attaque est d'envoyer un grand nombre de requêtes DNS avec un sous domaine aléatoire afin que le serveur DNS contacte tous les serveurs d'autorité du domaine Akamai (2017). Le but est de saturer le serveur d'autorité afin de rendre le domaine inaccessible.

L'amplification DNS (1.9) est aussi utilisée par quelques logiciels malveillants. Le but est d'envoyer de petites requêtes DNS en se faisant passer pour la victime afin que le serveur DNS renvoie de grosses réponses à la victime et ainsi, saturer sa bande passante.

famille (2) :déni de service permanent (PDoS)

Ces dernières années, deux nouvelles attaques de déni de service sont apparues : le « Firewall DoS » ou déni de Parfe-Feu (2.2) et le « Physical DoS » (2.1). La première va ajouter des règles dans le pare-feu de la victime afin de rejeter tous les paquets entrants et sortants. Il en résulte que la victime ne peut plus se connecter à aucun réseau. La seconde vise à détruire physiquement l'objet en écrivant des données aléatoires dans la mémoire et en supprimant le système de fichier et le système d'exploitation.

Espionnage (3)

Ici, on considère deux types d'espionnage : le général et l'industriel. Le premier correspond aux fonctionnalités classiques d'espionnage, comme le changement de DNS (3.1), l'exfiltration (3.2) de données, les attaques de l'homme du milieu (3.4), l'obfuscation de trafic malicieux(3.8),ou encore l'exploitation d'autres appareils sur le réseau local (3.3). Pour la fonctionnalité 3.1 le ver va changer les paramètres DNS de sa victime afin de rediriger l'ensemble de ses requêtes vers un serveur DNS malicieux qui servira ensuite à rediriger sa victime vers des sites frauduleux. La fonctionnalité 3.2 permet au ver de collecter des données du système de sa victime et de les envoyer à un serveur central appartenant à l'attaquant. La fonctionnalité 3.8 permet par exemple, de chiffrer le flux de données volées et envoyées au serveur central. Ici, l'espionnage industriel correspond aux fonctionnalités spécifiques à l'espionnage de sites industriels ou de grandes entreprises. Ces fonctionnalités sont : la mise en place d'un VPN inversé (3.7) pour accéder au réseau interne de l'entreprise, la cartographie de réseau local (3.6) ainsi que la surveillance et la prise de contrôle de système industriel SCADA (3.5).

Famille (4) Rentabilité

Cette famille rassemble toutes les fonctionnalités permettant de générer un revenu directement, sans avoir à louer les capacités du réseau de zombies. Aujourd'hui, il existe deux grandes familles de fonctionnalités permettant de générer un revenu actif : le minage de cryptomonnaies et la fraude au clic.

La première catégorie va voler la puissance de calcul des objets infectés pour leur faire miner des monnaies tels que le Litecoin (4.2), DogeCoin(4.1) ou l'Ethéréum. Ici, on peut rajouter un taxon pour chaque monnaie minée. La fraude au clic a pour but de simuler des clics sur un site web afin de faire monter le nombre de visites et ainsi empocher les revenus publicitaires

CloudFlare (2020). Pour cela, on peut rediriger les zombies vers le site en question. Une autre technique aurait été mise en place par le ver TheMoon au début de l'année 2019 Labs (2018). Cette dernière consiste à envoyer les zombies vers un ensemble de vidéos YouTube prédéfini afin de faire grimper le nombre de vues et donc générer plus de revenus publicitaires. Cependant, ces dernières fonctionnalités n'ont pas été observées dans les programmes étudiés. Elles ne sont donc pas incluses dans cette version de la taxonomie, mais le seront dans une prochaine version, après avoir trouvé et analysé plus de données sur les programmes malveillants découvert après 2018.

Famille(5) : Méthode d'exploitation

Afin de se propager et de gagner en puissance, un réseau de zombies doit exploiter des vulnérabilités dans un ensemble d'appareils vulnérables. Au cours de l'étude, nous avons remarqué que plusieurs méthodes existaient. La principale est l'utilisation d'un dictionnaire de mot de passe (5.1) afin de prendre le contrôle de l'objet via Telnet ou SSH. Mirai a introduit une nouvelle forme d'attaque en utilisant un dictionnaire pondéré (5.2). Ce dernier contient une liste de soixante-deux couples identifiants/ mots de passe, tous pondérés par une probabilité fixe. À chaque tentative, Mirai va sélectionner au hasard et en fonction des pondérations, dix couples à tester. Cela lui permet de garder une grande espérance d'exploitation tout en améliorant la vitesse de l'attaque.

En plus des attaques par dictionnaire, on peut observer l'utilisation ponctuelle d'exploitation de vulnérabilité commune « Common Vulnerability Exposure » (CVE) (53). Certains réseaux de zombies arrivent à en utiliser une tandis que d'autres peuvent en utiliser plusieurs (5.4).

Famille (6) : Architecture des victimes

Les objets connectés utilisent des architectures matérielles différentes et de ce fait, les réseaux de zombies doivent adapter leur programme en fonction. Ainsi, au début de l'utilisation de ce genre de programme malveillant, seules les architectures MIPS et MIPSEL (6.1) étaient supportées. En effet, les réseaux de zombies se propageaient via l'utilisation de binaires compilés uniquement pour ces architectures. Plus tard, des fonctionnalités de cross compilations ont été utilisées pour que plusieurs binaires soient envoyés et testés. Ainsi, les architectures ARM (6.2) et x86/64 (6.3) sont devenues exploitables. Plus tard encore, les vers ont développé une fonctionnalité pour se propager à l'aide de scripts BASH (6.4), infectant tout appareil disposant de cet interpréteur de commande, quelle que soit l'architecture de son processeur.

Famille (7) : Stratégie de Scan

Afin de trouver des cibles potentielles, un réseau de zombies doit constamment scanner le réseau Internet. Ainsi, il existe plusieurs stratégies, telles que le scan séquentiel d'une liste prédéfinie (7.1) ou d'un réseau (7.2) ou encore le scan aléatoire (7.3). La méthode de scan séquentiel peut être distribuée ou non. Dans le cas où elle est distribuée, chaque zombie nouvellement crée reçoit un sous-ensemble des adresses IP, qu'il scannera en ordre ascendant. La première stratégie a été implémentée par Hydra et l'attaquant devait donner une liste de cibles à scanner. Les réseaux de zombies suivants ont implémenté une nouvelle fonctionnalité rendant possible de donner un réseau ou un sous-réseau en argument et chaque zombie scannait l'ensemble du réseau de manière séquentielle.

Ensuite est apparu le scan aléatoire (ou « random scan »). Ici, le but est de scanner tout l'Internet en générant des adresses IP aléatoires. Cette stratégie est beaucoup plus efficace que les précédentes. Enfin, Mirai a introduit une nouvelle méthode pour scanner une victime sans

attendre sa réponse pour lancer le scan de la prochaine victime. En général les scans employés utilisent le « three hand checking » de TCP. Ici, Mirai contourne cette partie du protocole pour pouvoir scanner plus vite l'ensemble du réseau Internet. Les chercheurs appellent cette fonctionnalité « stateless scan » ou scan sans état (7.4). Cette fonctionnalité permet d'accélérer n'importe quelle stratégie de scan.

Famille (8) : Architecture du réseau de zombie

Les réseaux de zombies sont organisés en trois grandes familles : centralisée, décentralisée et hybride. La première famille correspond aux architectures où l'équipe d'attaquants met en place un serveur de contrôle central, où chaque zombie va s'enregistrer et duquel il va recevoir les ordres. Ce serveur est souvent appelé CC ou C2 pour « Command & Control »Il existe plusieurs méthodes de communications pour ces architectures : l'utilisation du protocole IRC (8.1), du protocole HTTP voir dans de rares cas, HTTPS ou la création d'un protocole de communication dédié comme le fait Mirai (8.2). Dans les réseaux de zombies que nous avons étudiés, seules les communications via IRC ou via un protocole binaire dédié ont été observées. C'est pour cela que les autres fonctionnalités, bien qu'observées dans d'autres réseaux de zombies, seront rajoutées à la taxonomie plus tard.

Pour la famille décentralisée, les réseaux sont en fonctionnement pair-à-pair (P2P) (8.2) et les protocoles de communication sont souvent dérivés du protocole BitTorent. Dans ce type de réseaux, il n'existe pas de serveur central et les commandes doivent se propager de proche en proche dans le réseau. On observe cependant plusieurs types de réseaux décentralisés en fonction du protocole qu'ils utilisent ou de leur structure. Ces taxons sont repris de l'étude faite part Rawat *et al.* (2018). De même pour la dernière famille d'architecture, les réseaux hybrides, utilisant réseaux pair-à-pair et serveurs de commande et de contrôle. Ici aussi, on ne prend en compte pour cette taxonomie que les réseaux P2P dérivant les protocoles BitTorrent et

muTorrent pour communiquer. Les deux seuls réseaux étudiés présentant cette fonctionnalité sont Wifatch et Hajime.

Famille (9) : Intelligence du programme

Dans cette dernière catégorie, nous retrouvons les fonctionnalités permettant d'améliorer le réseau de zombies de manière générale. Ainsi, on va par exemple retrouver les générateurs de nom de domaines (DGA) (9.3) permettant de cacher un serveur derrière un nom de domaine aléatoire, changeant tous les jours. Cette fonctionnalité permet de rendre plus difficile le travail des autorités et donc de faire survivre le réseau de zombies plus longtemps. Les fonctionnalités de modularité du programme malveillant et ses capacités de mise à jour (9.1) sont aussi décrites dans cette famille. Ces fonctionnalités permettent de faire rapidement évoluer le programme, pour le rendre plus efficace, plus discret ou pour lui donner de nouveaux buts.

Enfin, on retrouve les fonctionnalités de détection des architectures des victimes (9.2). Cette fonctionnalité permet au programme malveillant de ne transmettre que sa version compilée pour l'architecture de la cible. En effet, au début, les programmes envoyaient à la suite, tous leurs programmes et essayaient de les exécuter, jusqu'à ce que l'un fonctionne. Le fait de détecter et de n'envoyer que le bon binaire, permet de réduire les communications réseaux, donc d'être plus discret et plus rapide.

Famille (10) : Amélioration du temps de vie

Dans cette famille, nous retrouvons l'ensemble des fonctionnalités permettant d'augmenter significativement le temps de vie d'un programme malveillant dans son hôte. Cela se traduit par des fonctionnalités de persistance (10.1) et de mécanisme d'anti-redémarrage (10.2). La persistance peut être faite en modifiant le mircologiciel de l'objet, le programme malveillant

devient ainsi un élément à part entière du système. Le mécanisme d'anti-redémarrage consiste à tuer le chien de garde (watchdog) et ainsi empêcher les redémarrage automatique. Cette fonctionnalité est très utile, car la majorité des programmes malveillants ciblant les objets connectés réside en RAM et un simple redémarrage permet de supprimer l'infection.

Famille (11) : Prévention

L'ensemble des fonctionnalités de cette famille a pour but de rendre le réseau de zombies plus efficace dans son recrutement et d'affaiblir ses concurrents. On va donc retrouver les fonctionnalités de patch d'exploit utilisé, par exemple la fermeture des ports attaqués (11.2). Le but est ici d'éviter la surinfection de l'objet par d'autres réseaux de zombies ou par d'autres réplicats. On observe aussi des fonctionnalités plus avancées, comme la détection d'autres programmes malveillants et leur suppression (11.1). On a pu observer ce genre de comportement chez Wifatch par exemple.

Famille (12) : Discrétion

Cette famille de fonctionnalité permet aux réseaux de zombies d'être plus difficilement repérables, par les systèmes de détection d'intrusion (IDS), par les télescopes réseau ou par les pots de miel. On va ainsi retrouver des fonctionnalités pour supprimer le binaire du botnet (12.2) et faire en sorte qu'il ne réside qu'en RAM ou encore, pour stopper un processus comme Telnetd et prendre son nom (12.1). Enfin, on retrouve l'évasion de systèmes virtualisés (12.3) utilisée par Amnésia. Ici, le but est d'essayer de détecter lorsque le ver a infecté un pot de miel. Ainsi, le programme malveillant pourra se supprimer tout seul afin d'éviter que les équipes de chercheurs puissent l'étudier.

Fonctionnalité	Programme	Fonctionnalité	Programme
Syn Flood (1.1)	1, 2, 3, 4, 5, 9, 10, 12, 13, 15	UDP Flood (1.2)	2, 3, 4, 9, 10, 12, 13, 15
ICMP Flood (1.3)	2	ACK Flood (1.4)	3, 4, 5, 9, 10, 12, 13, 15
Push flood (1.5)	4, 10, 12, 13, 15	HTTP Flood (1.6)	4, 9, 10, 12, 13, 15
TCP XMAS (1.7)	4, 10	DNS Waterbording (1.8)	12, 13, 15
DNS Amplification (1.9)	12, 13, 15	Suppression de la mémoire (2.1)	14, 16
déni de Pare-feu (2.2)	16	Changement DNS (3.1)	3, 4
Exfiltration de données (3.2)	16	Exploitation d'appareil terminaux (3.3)	16
Attaque Homme du Milieu (3.4)	16	Surveillance de système SCADA (3.5)	16
Cartographie de réseau local (3.6)	16	VPN inversé (3.7)	16
Obfuscation de trafic malicieux (3.8)	16	Minnage de DogeCoin (4.1)	7
Minnage de LiteCoin (4.2)	7	Attaque par dictionnaire (5.1)	1, 2, 3, 4, 5, 7, 8, 9, 10, 11
Attaque par dictionnaire pondéré (5.2)	12, 14	Exploitation d'une CVE (5.3)	7, 9, 13
Exploitation de plusieurs CVE (5.4)	15, 16	MIPS/EL(6.1)	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 16
ARM (6.2)	5, 6, 7,8, 9, 10, 11, 12, 16	x86/64 (6.3)	5, 6, 7, 8, 11, 12, 16
BASH (6.4)	14, 16	Liste prédéfinie (7.1)	1
Scan de réseau séquentiel (7.2)	2, 3, 4	Scan aléatoire (7.3)	5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
Scan sans état (7.4)	12, 14, 16	Architecture Centralisé (IRC CC) (8.1)	1, 2, 3, 4, 5, 7, 10, 13
Architecture P2P (Décentralisée) (8.2)	8, 11	Architecture Centralisé (protocole dédié) (8.3)	6, 9, 12, 14, 15, 16
Modularité du code ou système de MAJ (9.1)	11, 15, 16	Détection de l'architecture de la victime (9.2)	10, 11, 12, 15, 16
Algorithme DGA (9.3)	12, 16	Persistance (10.1)	16
Anti-redémarrage (10.2)	11, 12, 15, 16	Suppression d'autres programmes (11.1)	5, 7, 8, 11, 12, 13, 15
Fermeture de ports (11.2)	4, 5, 7, 8 9, 11, 12, 13, 15	Vol de nom de processus (12.1)	10, 11, 12, 15
Suppression du binaire (12.2)	11, 12, 15	évasion de systèmes virtualisés (12.3)	13

Tableau 3.5 – Fonctionnalités des programmes malveillants et de leur réseaux de zombies

3.3.2 UNE NOUVELLE REPRÉSENTATION DE L'ÉVOLUTION DES LOGICIELS MALVEILLANTS

Afin de représenter l'évolution des programmes malveillants infectant les objets connectés, nous avons développé deux graphes. Le premier que nous appelons graphe phylogénique permet de représenter le nombre de points communs entre deux programmes et donc de possibles liens de parenté. Pour ce faire, nous déterminons les fonctionnalités implémentées par chaque programme.

Ensuite, nous déterminons le nombre de fonctionnalités communes entre les programmes. Chaque programme possède un cercle dont le rayon est proportionnel à son nombre de fonctionnalités total. Nous traçons ensuite un lien entre deux programmes, pondéré par le nombre de fonctionnalités communes. Cependant, il peut y avoir des doublons et afin de rendre le graphe plus lisible, nous les supprimons.

Pour ce faire, si deux chemins existent entre plusieurs programmes, nous ne gardons que le plus long. Par exemple, dans notre graphe donné en figure 3.2, Hydra possède 4 points communs avec Chuck Norris et 4 avec Psybot. De plus, Psybot possède 6 points communs avec Chuck Norris. De ce fait, les fonctionnalités communes entre Hydra et Chuck Norris, sont aussi communes avec Psybot et il y a plus de chance pour que Chuck Norris se soit inspiré de Psybot que d'Hydra. Ainsi, nous supprimons le lien entre Hydra et Chuck Norris.

Dans ce cas précis, plusieurs chercheurs pensent que les auteurs de Psybot et ceux de Chuck Norris sont les mêmes personnes, confirmant nos hypothèses Čeleda *et al.* (2010). Cependant, ce n'est pas forcément le cas pour tous les liens entre programmes malveillants. De plus, nous pouvons observer que notre méthode ne permet pas de supprimer tous les cycles du graphe. Enfin, cette représentation manque de précision et ne permet pas de faire ressortir les fonctionnalités qui se propagent ainsi que les programmes les ayant introduites.

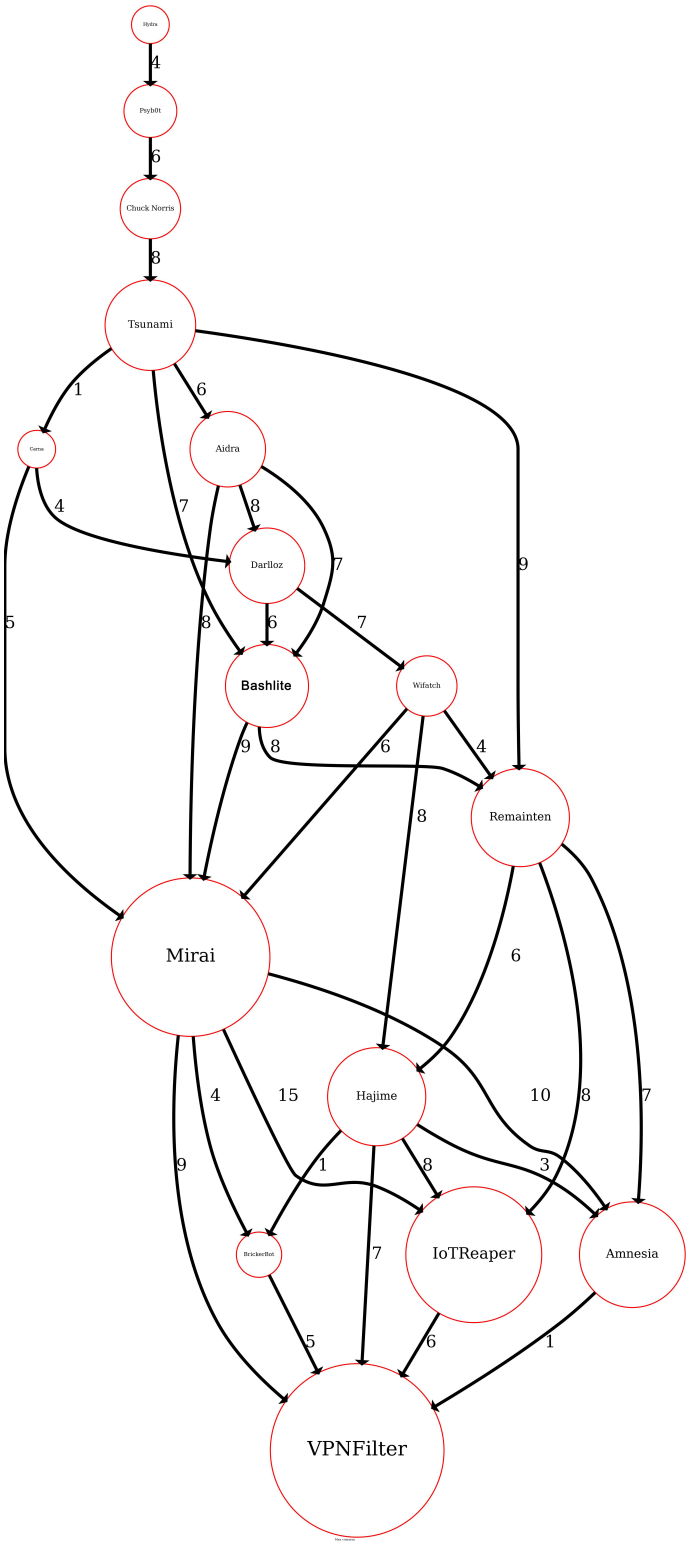

Figure 3.2 – Graphe Phylogénique des programmes malveillants.

Afin de pallier à ces défauts, nous avons mis en place une autre forme de représentation, plus simple et plus intuitive, que nous appelons graphe de propagation de fonctionnalités. Ici, nous attribuons une couleur et un style (trait plein, pointillé, etc) à chaque fonctionnalité. Nous traçons ensuite un cercle de cette couleur et de ce style autour du noeud du premier programme observé utilisant cette fonctionnalité. Enfin, nous traçons un lien entre ce programme et tous les suivants utilisant cette fonctionnalité. Les couleurs et les styles permettent de facilement différencier les fonctionnalités.

Cette représentation permet de rapidement voir quelles sont les fonctionnalités les plus utilisées et les plus transmises. Cela permet aussi de voir quels sont les programmes les plus innovants, ayant introduit de nouvelles fonctionnalités. Nous avons donné un exemple en Figure 3.3, représentant les transmissions des fonctionnalités relatives aux objectifs.

On peut observer sur cette image que pour la majorité des fonctionnalités d'attaques par déni de services distribués, les fonctionnalités sont introduites par un programme et transmises à la totalité des vers suivants. C'est le cas par exemple, de la fonctionnalité d'attaque SYN flood, mise en place par Hydra et transmise à tous les réseaux de zombies ayant pour objectif de créer des attaques de déni de service.

L'autre avantage d'utiliser cette représentation est de pouvoir déterminer rapidement diverses familles de réseaux de zombies. En effet, on peut observer sur notre figure 3.3 quatre grandes familles. La première, la plus grande, est celle des réseaux ayant pour objectif de créer des attaques de déni de service permanent. La seconde, composée ici d'un seul vers, est la famille des réseaux minant des cryptomonnaies.

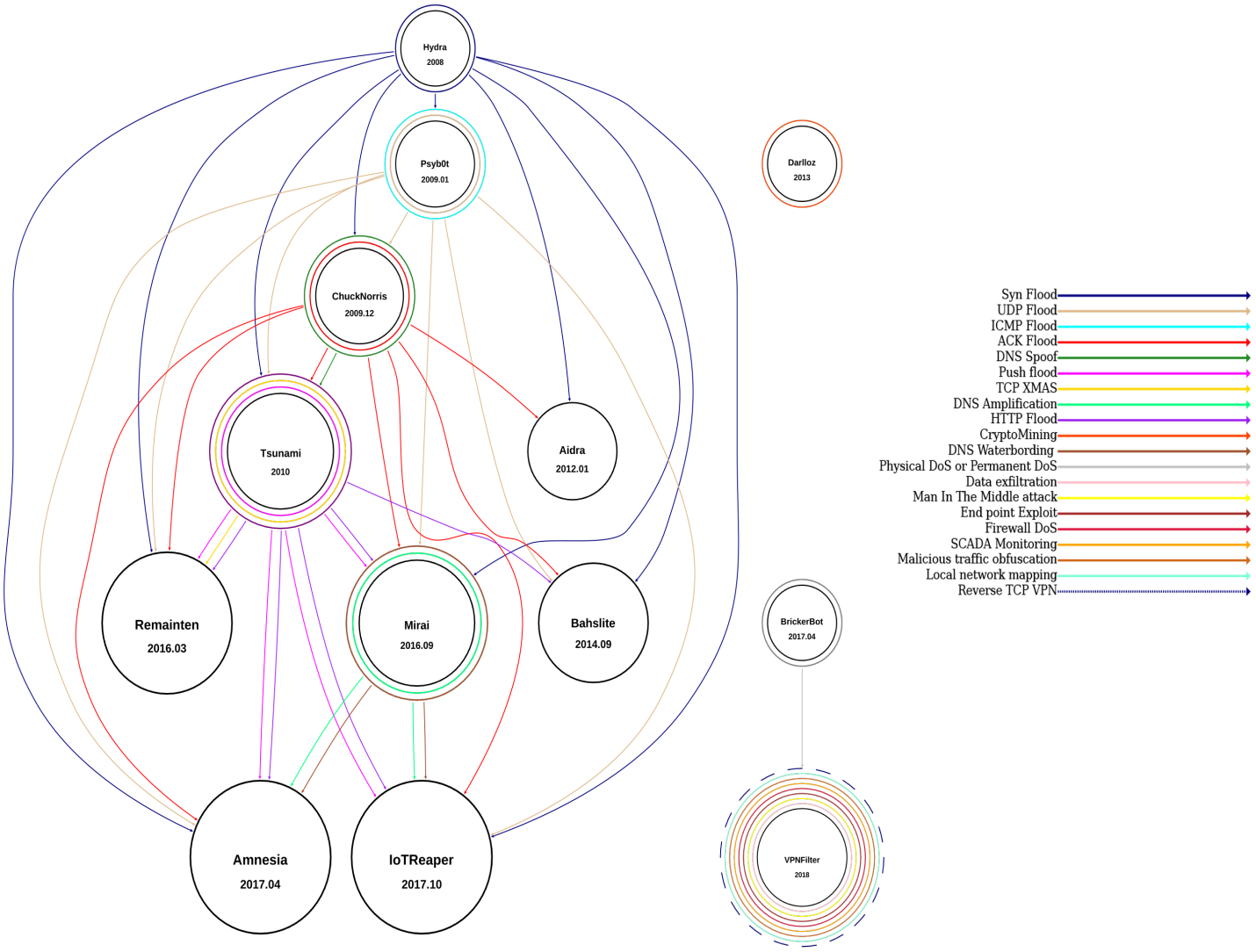

Figure 3.3 – Propagation des fonctionnalités d'attaque.

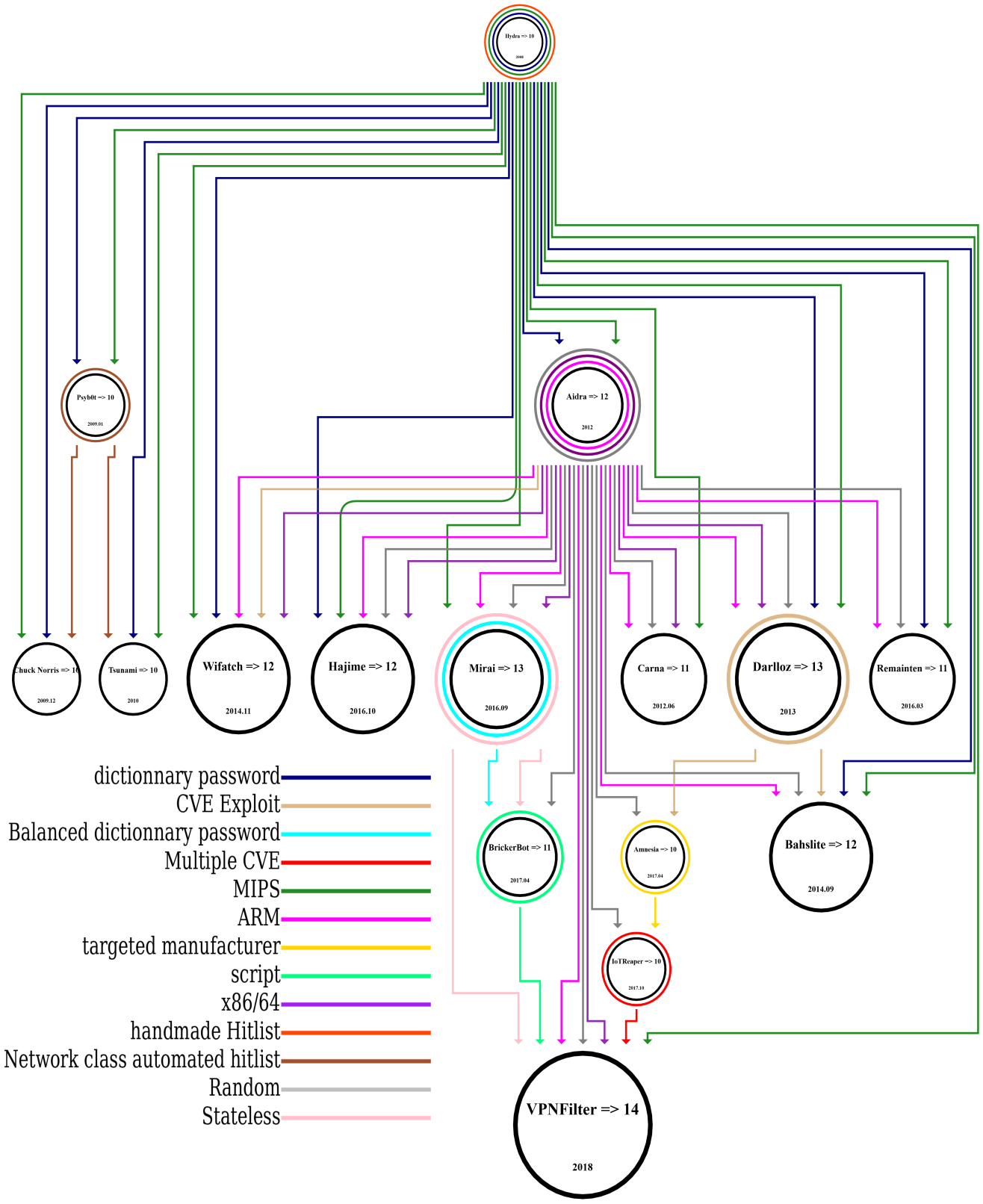

Figure 3.4 – Propagation des fonctionnalités d'infection.

Enfin, on observe la famille des réseaux ayant pour objectif de mettre en place des déni de service physique, composé de BrickerBot et de VPNFilter. Ce dernier, compose aussi à lui seul une autre famille, celle des vers-espions, dont l'objectif est de voler un maximum de données, à des particuliers ou à des industriels. On observe aussi que parmi tous les réseaux de zombies étudiés ici, il a introduit la totalité des fonctionnalités associée à cette famille.

Cette taxonomie et ces nouvelles représentations nous permettent d'observer les évolutions au sein des réseaux de zombies et de leurs programmes malveillants ciblant les objets connectés. En connaissant l'évolution des réseaux de zombies d'objets connectés, on peut déterminer les fonctionnalités les plus présentes, les plus efficaces, et donc ayant le plus de chance d'être réutilisées par de futurs réseaux de zombies. Avec ces connaissances, on pourrait adapter les efforts de recherches afin de se focaliser en priorité sur la détection de ces fonctionnalités, ainsi que dans la création de réponses adaptées permettant de réduire la nuisance des réseaux de zombies.

Cependant, afin de pouvoir mieux comprendre ces évolutions, nous avons besoins d'outils afin de modéliser et comprendre les impacts de chaque fonctionnalités. C'est pourquoi nous détaillerons dans le chapitre suivant la création d'un outil de simulation, ainsi que les expériences faites avec.

CHAPITRE 4

OUTILS DE SIMULATION D'INFECTION ET DE PROPAGATION D'ÉPIDÉMIE

Au cours de ce chapitre, nous allons présenter les divers modèles et outils décrivant les propagations d'agents pathogènes dans une population. En effet, ce problème, à l'origine médical, est extrêmement proche de notre problématique et plusieurs équipes de chercheurs les ont utilisés afin de décrire l'évolution de la taille de réseaux de zombies. Nous y détaillerons ensuite notre modèle et le logiciel que nous avons développé. Puis, dans une seconde partie, nous présenterons notre méthodologie au travers des divers paramètres utilisés pour nos expériences. Nous décrirons ainsi les diverses simulations effectuées. Enfin la quatrième section présente les résultats de chaque simulation. Le but de créer un outil de simulation modélisant les populations de réseaux de zombies, est de pouvoir simuler certains comportements clés impactant la vitesse de propagation et la taille du réseau de zombies. Grâce à cela, nous pouvons exploiter au mieux notre taxonomie, en déterminant quelles fonctionnalités sont les plus performantes, et donc axer les travaux de défense sur ces dernières qui auront une plus grande probabilité d'être utilisées par les réseaux de zombies.

4.1 LES MODÈLES DE SIMULATIONS D'INFECTIONS DE RÉSEAUX DE ZOMBIES

Dans cette section nous allons analyser les outils existants permettant de modéliser la propagation d'un ver informatique. Nous allons ensuite détailler le programme que nous avons créé afin modéliser la propagation d'un ou de plusieurs vers en concurrence. Enfin, nous expliciterons nos résultats.

4.1.1 LES MODÈLES EXISTANTS

Nous avons montré dans le précédent chapitre que les programmes malveillants qui créent des réseaux de zombies évoluent. Nous avons également montré que certaines fonctionnalités tendent à disparaître alors que d'autres tendent à perdurer. Nous émettons l'hypothèse que les fonctionnalités qui apportent plus d'avantages pour le réseau auront tendance à survivre. Ainsi, notre objectif est de modéliser l'impact que peuvent avoir diverses fonctionnalités sur l'efficacité d'un réseau de zombies. Pour ce faire, nous observerons les incidences sur la taille du réseau (nombre de zombies) ainsi que sur la vitesse de propagation de l'infection. En effet, un réseau qui se construit plus vite et qui obtient plus de zombies sera plus efficace pour remplir ces objectifs, quels qu'ils soient. Enfin, nous modéliserons la concurrence entre divers réseaux afin de montrer l'impact réel de ces fonctionnalités.

Notre objectif étant de mesurer la taille d'un réseau et la vitesse de la propagation de l'infection, nous nous sommes tournés vers les modèles infectieux. Cependant, il existe d'autres modèles de propagation, analysés par Wainwright et Kettani (2019). Dans leur article, les auteurs font un état de l'art de l'ensemble des modèles de réseaux de zombies existants. Afin d'analyser les divers modèles existants, ils introduisent un cadre d'analyse basé sur le cycle de vie des réseaux de zombies. Pour ce faire, ils utilisent les critères suivants : Conception, Recrutement,

Interaction, Marketing et Exécution des attaques. Ils appellent ce cadre « CRIME ».

Un modèle décrivant la phase de conception devra lister les motivations de l'attaquant avec les fonctionnalités qu'il devra implémenter pour les satisfaire. La phase de recrutement correspond à l'ensemble des descriptions des méthodes de propagation et d'infection. Les interactions décrivent les communications internes ainsi que les méthodes de gestion du réseau de zombies. La phase de marketing correspond à l'ensemble des actions possibles que peut faire un attaquant pour vendre ou rentabiliser son réseau. Enfin, la partie exécution décrit les buts finaux du réseau, donc l'ensemble des attaques qu'il peut créer et leurs conséquences. Le but des auteurs est de déterminer, pour chaque modèle, quelles phases du cycle de vie il décrit.

Parmi les sept modèles analysés, seuls trois décrivent la phase de recrutement : les modèles infectieux, les modèles d'apprentissage machine et les modèles issus de la théorie des jeux. Le premier est dérivé des modèles épidémiologiques utilisés en médecine. Au départ, les modèles utilisaient un schéma S.I.C (Susceptible Infecté Contrôlé), utilisant une population homogène où tous les hôtes sont Susceptibles d'être infectés. Il existe une faible probabilité que chaque hôte devienne Infecté puis Contrôlé. Plus tard un autre modèle S.I.S (Susceptible Infecté Susceptible) est apparu pour modéliser l'utilisation de plusieurs méthodes d'exploitation par les réseaux de zombies. Ces modèles permettent donc de modéliser l'évolution de la population de zombies au cours du temps.

Le second modèle analysé est celui concernant l'apprentissage machine. Ces modèles se basent sur les données récoltées par les réseaux de pots de miels ainsi que sur les systèmes de détection d'intrusion (IDS). Leur but est de classer les données afin d'identifier le trafic malicieux d'un réseau de zombies au travers de l'ensemble des communications réseau enregistrées. Ces modèles se concentrent sur les phases de communication et de recrutement. Cependant, contrairement au premier modèle, ils ne permettent pas d'évaluer ou de prédire la taille d'un réseau de zombies.

Les modèles de la théorie des jeux évolutionnaire sont utilisés conjointement avec les modèles épidémiologiques afin d'analyser la persistance d'un réseau de zombies par rapport aux autres. Cela permet de décrire les interactions entre les possesseurs de ces réseaux, telles que la collaboration ou la compétition. Ces modèles discutent principalement des motivations et légèrement de la phase de recrutement. Ici aussi, les modèles ne permettent pas de prédire la taille ou la vitesse de propagation d'une infection.

De ce fait, nous nous sommes concentrés sur les modèles infectieux et leurs applications concernant les réseaux de zombies. Nous avons donc analysé quatre modèles épidémiologiques, en nous basant sur des articles les utilisant afin de modéliser la propagation de certains vers comme Code Red ou Morris (Zou *et al.*, 2002).

4.1.2 CRITÈRES D'ANALYSE DES MODÈLES EXISTANTS

Afin de les comparer, nous avons défini plusieurs critères. Le premier est le niveau de simplification du modèle (ou simplicité). Nous disons d'un modèle qu'il est simplifié s'il combine plusieurs facteurs en un seul afin de simplifier les modélisations et les calculs. Un tel modèle peut difficilement décrire les impacts que pourrait avoir une fonctionnalité sur l'efficacité d'un réseau de zombies.

Notre second critère est l'extensibilité du modèle. Un modèle extensible pourra prendre en compte facilement l'ajout de nouvelles fonctionnalités. Par exemple, si une nouvelle méthode de détection des victimes apparaît, le modèle extensible pourra la décrire facilement sans avoir à modifier sensiblement le modèle. Un outil non extensible engendrera une grande complexité et un besoin en calcul important pour décrire un nouveau comportement. Enfin, nous déterminerons si les modèles existants permettent de décrire la compétition entre deux réseaux de zombies.

4.1.3 ANALYSE DES MODÈLES SÉLECTIONNÉS

Les premiers modèles apparus sont les modèles S.I (Susceptible Infecté), S.I.R (Susceptible Infecté Rétabli) et S.I.S (Susceptible Infecté Susceptible). Ces modèles se basent sur l'état des individus d'une population pour modéliser la propagation d'une épidémie. Dans le modèle S.I.R, les ordinateurs vulnérables à un ver sont susceptibles, lorsqu'ils sont exploités ils deviennent infectés et lorsque le ver est supprimé ils deviennent rétablis. Dans le modèle S.I.S, les individus peuvent redevenir Susceptible si la vulnérabilité n'est pas corrigée ou si le ver utilise plusieurs méthodes de contamination. Dans une population donnée, tous les agents ne sont pas forcément susceptibles. La taille maximale de l'épidémie est donc déterminée par la proportion de machines vulnérables.

Dans ces modèles, on utilise un taux de naissance ou taux de propagation, souvent constants pour décrire la vitesse de propagation. On utilise aussi un taux de mort pour représenter les individus qui sortent temporairement du réseau (ordinateur éteint, virus supprimé, etc.). Ce modèle est très simplifié, il néglige les changements d'états internes de l'individu, abstrait le phénomène de transmission, d'infection et de récupération par les taux de naissance et de mort. Or, ces taux sont influencés par de nombreux paramètres comme la méthode de scan, la congestion du réseau, le nombre de machines qui se connectent ou se déconnectent, etc. Le modèle S.I est une version encore plus simplifiée du modèle S.I.S où l'on ne prend pas en compte la perte d'hôtes infectés.

Ces modèles sont très utilisés pour modéliser la propagation de vers informatiques utilisant une stratégie de détection aléatoire (Zou *et al.*, 2002; Staniford *et al.*, 2002; Moore *et al.*, 2003). En effet, ils supposent qu'une certaine proportion de la population est vulnérable au ver et que ce dernier va se propager aléatoirement au cours du temps. Chaque nouvel hôte va attaquer au hasard un individu de la population afin d'essayer de propager l'infection. Dans l'ensemble, ces modèles utilisent des équations différentielles pour abstraire les comportements et modéliser la

propagation de l'infection.

Dans l'ensemble de ces modèles, on ne considère pas le phénomène de patch de vulnérabilité, car la propagation d'une épidémie est bien plus rapide que ne peut l'être une réponse technique permettant de rapidement désinfecter les ordinateurs corrompus. De plus, ces outils sont très spécifiques à certaines infections et modélisent très mal certains comportements. C'est ce que montre l'équipe de Chen et Ji (2005). En effet, ils souhaitent modéliser la propagation de ver utilisant la structure du réseau pour trouver de nouvelles victimes. D'après eux, il est très difficile de le faire en utilisant simplement les modèles S.I.S classiques, car dans une propagation topologique, le taux d'infection est différent pour chaque noeud. Ainsi, ils développent un modèle basé sur les chaînes de Markov afin de mieux décrire ces phénomènes. L'équipe de Zou *et al.* (2002) en a fait de même pour correctement simuler l'attaque du ver Code Red. Ils ont rajouté plusieurs fonctions au modèle de base afin de décrire la chute du taux d'infection du vers. Ils ont ainsi créé une nouvelle version du modèle S.I.R, plus complexe et plus précise que la version basique.

Un modèle un peu plus proche du fonctionnement de ces vers est celui décrit par (Chen *et al.*, 2003). Dans leur article, ils définissent un nouveau modèle de propagation des vers informatique, appelé AAWAP et utilisant le scan aléatoire. Pour ce faire, ils se basent sur les épidémies engendrées par les vers Morris, Code Red et Nimba. Contrairement aux modèles épidémiologiques, ils utilisent un système discret se basant sur l'état de la population à un temps t pour déterminer l'état probable de la population à un temps $t+1$. De plus, ils prennent en compte le taux de correctif (patch). Pour modéliser un comportement de scan topologique, ils divisent l'espace des machines en sous-espaces, de tailles différentes et ayant chacun une probabilité d'être choisies. Cela leur permet d'étendre facilement leur modèle. Cependant, les auteurs combinent l'ensemble des fonctionnalités de scan et d'infection en un taux de scan, représentant le nombre de machines que peut scanner un zombie par unité de temps.

On voit ici que ces modèles sont très simplifiés, ils fusionnent l'ensemble des comportements de scan et d'infection en un seul paramètre. De plus, ils ne sont pas (ou peu) extensibles ; l'ajout d'un nouveau comportement oblige à modifier profondément le modèle. Enfin, ces outils ne sont utilisés que pour décrire la propagation d'un seul ver. Si l'on souhaite modéliser le phénomène de concurrence, il faudrait redéfinir les équations utilisées, car chaque réseau peut influencer les autres de manière significative.

4.1.4 ZOMBOTS : LE JEU DE SIMULATION DES PANDÉMIES DE ROBOTS ZOMBIES

Ainsi afin d'observer les impacts de chaque fonctionnalité sur l'efficacité d'un réseau de zombies, nous avons souhaité créer notre propre outil. Nous avions besoin d'un outil facilement extensible et très proche du fonctionnement réel des vers à étudier. Ainsi, plutôt que de définir des équations différentielles ou des suites arithmétiques, nous avons développé un programme de simulation sous forme d'un jeu tour par tour.

Pour fonctionner, il faut définir la machine d'état représentant le cycle de vie d'un zombie. Chaque état possède un nombre de tours qui lui est propre. Cela permet de représenter le temps que met le zombie pour accomplir une tâche. Ce temps peut être fixe ou déterminé à l'aide d'une fonction mathématique si ce dernier est variable. Cela peut être utile pour simuler la latence d'un réseau ou sa congestion progressive. Certains états sont obligatoires, comme par exemple, le scan et l'exploitation. L'état de scan permet de générer l'adresse IP à scanner en fonction de la stratégie employée. Une fois l'adresse générée, l'environnement de jeu va déterminer s'il correspond à une adresse vulnérable ou non. Il va ensuite transmettre le résultat au zombie qui pourra passer en état d'exploitation ou retourner en phase de scan. Cet automate d'état est représenté en figure 4.1.

Pour initialiser la simulation, on crée un nombre aléatoire de victimes (en utilisant une proportion

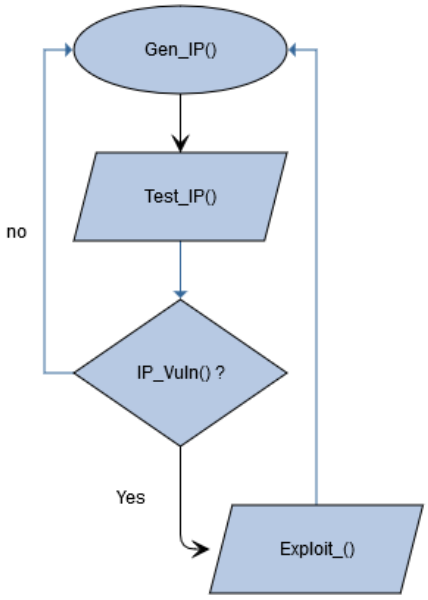

Figure 4.1 – Fonctionnement général d'un zombie lors du processus d'infection

donnée) vulnérable à chaque type de réseau de zombie. Ensuite, on va faire jouer chaque zombie dans un ordre aléatoire défini au début de chaque tour. Pour cela, on les fait avancer à l'état suivant. Si le zombie a généré une adresse IP, l'environnement va vérifier si cette dernière est vulnérable ou non. Si tel est le cas, on ajoute un zombie aux réseaux qui l'a infecté et les deux zombies passent en phase d'exploitation. L'état de la victime peut aussi être modifié si le programme malveillant corrige les failles qu'il a utilisées afin d'éviter que d'autres réseaux ne puissent acquérir la victime. Enfin, si un ver supprime les autres vers ayant déjà infecté la victime, on supprime un zombie dans chacun de ces réseaux. L'ensemble de cette phase de jeu est répété autant de fois que nécessaire. Un schéma récapitulatif est donné en figure 4.2. Sur cette figure, l'étiquette *GenIP()* correspond à la phase de génération de l'adresse IP qui sera attaquée, l'étiquette *TestIP()* correspond à la phase de détection de vulnérabilités exploitables chez la victime. Si une vulnérabilité est exploitable, alors on rentre dans la phase d'exploitation de la victime, étiquetée *Exploit()*.

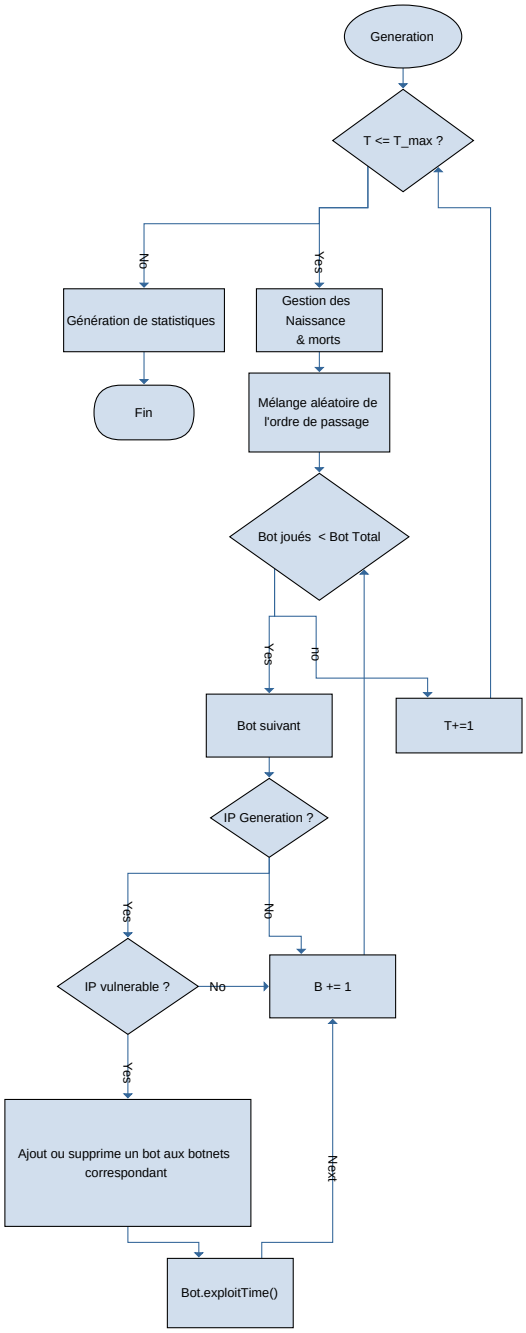

Figure 4.2 – Fonctionnement général du jeu

Ainsi, notre outil permet de très facilement observer les impacts que peuvent avoir chaque fonctionnalité sur l'efficacité du réseau de zombies. De plus, nous pouvons aisément simuler

une concurrence entre les réseaux de zombies par exemple entre Wifatch et ses prédécesseurs. La mort et la naissance de nouveaux appareils peuvent aussi être ajoutées en début ou en fin de tour. Pour ce faire, il suffit de déterminer un nombre aléatoire de machines à supprimer (et de les supprimer de leur réseau de zombies si elles ont été infectées) et à en générer de nouvelles. Là aussi, le nombre peut être fixe ou dépendre d'une fonction plus précise.

Le nombre d'états d'un zombie ainsi que les méthodes pour déterminer les temps pris par chaque phase ne sont pas limités. Ainsi, notre modèle ne fusionne pas divers comportement ou fonctionnalité sous un seuls paramètre, car on peut facilement modéliser chaque fonctionnalité et observer leur impact. Il est très extensible de par sa conception. Enfin, notre modèle simule la concurrence entre plusieurs réseaux de zombies (similaires ou différents). L'ensemble de cette partie est récapitulé dans le tableau 4.1. Pour rappel le critère de simplicité représente le fait que le modèle fusionne plusieurs comportements en un seul paramètre, l'extensibilité correspond à la capacité du modèle à pouvoir modéliser de nouvelles fonctionnalités, et la concurrence représente la capacité du modèle à décire la concurrence entre réseaux de zombies.

Modèle	Simplicité	Extensibilité	Concurrence	Type
S.I	✓	✗	✗	Déterministe
S.I.S	✓	✗	✗	Déterministe
Modèle Markovien	✓	✓	✗	Stochastique
AAWAP	✓	✓	✗	Stochastique
Notre modèle	✗	✓	✓	Stochastique

Tableau 4.1 – Comparaison des différents modèles

4.2 LES SIMULATIONS EFFECTUÉES AVEC NOTRE MODÈLE.

Dans cette section nous allons détailler les paramètres utilisés pour faire nos expériences. Nous allons ensuite exposer et analyser nos résultats expérimentaux.

Afin d'observer les différents impacts de quelques fonctionnalités, nous avons mis en place quelques simulations. Notre but est ici d'observer l'impacte de la stratégie de recherche de victimes (famille numéro 7 de notre taxonomie) sur la vitesse de propagation d'un réseau de zombies. Ensuite nous avons souhaité étudier l'impact des fonctionnalités de prévention des autres réseaux de zombies (famille numéro 11) sur l'efficacité d'un réseaux de zombies.

Pour chaque expérience, nous avons fait 105 simulations et avons tracé pour chaque réseau de zombies, sa population maximale, minimale, moyenne et médiane des simulations. Nous avons ensuite mis sur un même graphique les populations médianes de chaque réseau afin de les comparer plus facilement.

Les taux de mort et de naissance suivent une loi normale de moyenne 5 et d'écart-type 2. Le taux de mort a une fréquence d'apparition de 200 tours, tandis que le taux de naissance possède une fréquence d'apparition de 150 tours. Cela permet de modéliser simplement l'ajout et le retrait de nouveaux équipements connectés. Comme le nombre d'objets connectés est en hausse depuis plusieurs années, nous avons choisi de mettre en place une fréquence d'apparition plus élevée pour le taux de naissance que pour le taux de mort. Cependant, les valeurs sont ici arbitraires, car nous souhaitons observer les impacts des fonctionnalités et non modéliser les comportements réels d'un réseau de zombies particulier. Ces valeurs et ces fonctions peuvent être modifiées aisément si l'on souhaite reproduire la propagation d'une infection particulière.

Ici le taux de naissance et le taux de mort sont les mêmes pour toutes les expériences, afin qu'ils n'influencent pas nos expériences. Cette fonctionnalité sera utile pour de futures extensions

de ces travaux, où l'on pourrait souhaiter, par exemple, modéliser de manière très fidèle la propagation d'une infection sur un temps cours, tout en prenant en compte l'apparition et la disparition d'objets connectés du réseau Internet.

De même la notion de tour est une abstraction du temps, ainsi un tour représente une quantité atomique de temps. La correspondance en secondes, minutes etc, dépend ainsi de l'expérience et de ce que l'on souhaite observer. Pour nos expériences, le temps est abstrait complètement, nous souhaitons observer les conséquences des écarts de temps consommé par chaque fonctionnalité. Qu'une fonctionnalité A prenne 40ms de moins que la fonctionnalité B pour effectuer la même tache ne nous importe pas ici. Ce que nous souhaitons observer sont les conséquences d'un écart de temps, et comment cet écart va impacter l'efficacité générale du réseau de zombies. Peu importe que l'écart soit de 40ms ou 2min.

La première expérience a pour but d'observer la concurrence entre plusieurs réplicats d'un même programme malveillant. Cela se produit lorsque le code source d'un programme permettant de créer un réseau de zombies est vendu à plusieurs entités ou mis à disposition en ligne (comme pour Mirai par exemple). On a ainsi une compétition entre chaque réseau de zombies afin de capturer un maximum d'hôtes. Pour ce faire, nous avons simulé un programme ayant un scan aléatoire des victimes et s'immunisant à l'ensemble de ses réplicats. Cela veut dire que lorsqu'il infecte une victime, le programme va corriger la faille lui ayant permis d'entrer. Un programme comme Mirai fait cela en fermant les ports Telnet et SSH.

Notre simulation se fait sur 1500 tours, avec une population totale de 100000 individus dont 30% sont vulnérables aux vers. Nous avons fait deux expériences avec deux et cinq réplicats. Nous avons ensuite fait une simulation où chaque réseau commence à un temps différent. L'ensemble des paramètres de chaque réseau est donné dans le tableau 4.2. Nous avons choisi ces paramètres de manière arbitraire, concernant la durée de la simulation (1500 tour) de manière a pouvoir observer la totalité de l'évolution du système et donc observer le moment où le système devient

stable. Nous avons remarqué que dans la majorité des expériences, au bout de 800 tours, le système était stable. Ainsi, 1500 tours nous a paru un bon compromis, permettant de nous assurer de la stabilité du systèmes. Cependant, nous avons quand même fait varier ces paramètres pour certaines expériences (sans varier les autres) afin de montrer la pertinence de ces choix.

Notre deuxième expérience compare les différentes stratégies de scan. Nous avons simulé deux réseaux de zombie ayant des caractéristiques identiques, sauf pour la méthode de génération d'adresses à tester. Le premier les génère de manière aléatoire, tandis que le second les génère de manière séquentielle. Cela correspond aux fonctionnalités 7.2 et 7.3 de notre taxonomie. Il commence donc par la 1^{re} adresse, puis la 2^{me}, etc. De plus, chaque zombie reprend le scan depuis le début. Ce comportement peut être observé sur le tout premier programme malveillant pour objet connecté Hydra. En effet, la méthode de scan n'était pas encore automatisée et chaque zombie scannait l'ensemble des adresses IP en partant du début.

Cette dernière méthode étant particulièrement inefficace, nous avons fait une simulation sur 1000 tours, puis une sur 5000 tours afin d'observer la totalité de la compétition entre les deux vers. Au départ nous souhaitions effectuer une simulation sur 100000 tours pour laisser le premier réseau scanner l'ensemble des adresses existantes. Cependant, au bout de 750 tours, nous avions remarqué que le système était stable et que les populations des réseaux n'évoluaient que très peu. Ici, les victimes étaient vulnérables aux deux réseaux. Il y avait 30% des individus de vulnérables aux deux réseaux. Nous avons ensuite refait l'expérience, en attribuant des temps d'action relativement plus faibles pour le réseau de zombies utilisant la méthode de scan séquentielle que pour le réseau utilisant le scan aléatoire. L'ensemble des paramètres de chaque réseau est donné dans les tableaux 4.3 et 4.4.

Enfin, nous avons voulu observer le phénomène de suppression des concurrents. Cela s'est vu avec Wifatch, qui supprimait l'ensemble des programmes malveillants qu'il trouvait chez ses victimes et les immunisait ensuite. Cela correspond aux fonctionnalité 11.1 et 11.2 de notre

taxonomie. Ici, nous avons fait une simulation avec des réseaux identiques, à la seule différence que le premier n'immunisait pas ses victimes, là où le second supprimait les infections du premier et immunisait ses victimes. Nous avons ensuite fait de même, mais en modifiant la stratégie de scan du premier réseau afin qu'il utilise une génération d'adresse séquentielle. Dans tous les cas, nous avons fait une simulation avec les deux réseaux actifs en même temps et une simulation avec le second réseau (celui supprimant les infections de l'autre) démarrant avec un retard de 200 tours de jeu. Ici, les simulations se faisaient sur 1500 tours avec 30% des individus vulnérables. L'ensemble des paramètres de chaque réseau est donné dans les tableaux 4.5 et 4.6. Dans ces tableaux, A signifie *Aléatoire* et correspond à la fonctionnalité 11.2 de notre taxonomie ; S signifie *Séquentiel* et correspond à la fonctionnalité 11.1.

Réseau de zombies	Méthode de scan	Temps de génération IP	Temps de test	Temps d'exploit	Suppression	Immunité	Départ (t)
botnet #1	A	3	5	4	∅	Tous	0
botnet #2	A	3	5	4	∅	Tous	100
botnet #3	A	3	5	4	∅	Tous	150
botnet #4	A	3	5	4	∅	Tous	300
botnet #5	A	3	5	4	∅	Tous	1000

Tableau 4.2 – Expérience 1 : concurrence entre réseau du même type

Réseau de zombies	Méthode de scan	Temps de génération IP	Temps de test	Temps d'exploit	Suppression	Immunité	Départ (t)
botnet #1	S	1	5	4	∅	#1 #2	0
botnet #2	A	3	5	4	∅	#1 #2	0

Tableau 4.3 – Expérience 2a : concurrence entre stratégie de scan aléatoire et séquentielle

Réseau de zombies	Méthode de scan	Temps de génération IP	Temps de test	Temps d'exploit	Suppression	Immunité	Départ (t)
botnet #1	S	1	1	1	∅	#1 #2	0
botnet #2	A	3	5	4	∅	#1 #2	0

Tableau 4.4 – Expérience 2b : concurrence entre stratégie de scan aléatoire et séquentielle

Réseau de zombies	Méthode de scan	Temps de génération IP	Temps de test	Temps d'exploit	Suppression	Immunité	Départ (t)
botnet #1	A	3	5	4	∅	∅	0
botnet #2	S	3	5	4	#1	#1 #2	200

Tableau 4.5 – Expérience 3a : concurrence, stratégie identique avec surpression

Réseau de zombies	Méthode de scan	Temps de génération IP	Temps de test	Temps d'exploit	Suppression	Immunité	Départ (t)
botnet #1	S	1	1	1	∅	∅	0
botnet #2	A	3	5	4	#1	#1 #2	200

Tableau 4.6 – Expérience 3b : concurrence, stratégie différente avec surpression

4.3 LES RÉSULTATS DES SIMULATIONS

Dans cette section nous allons présenter les résultats de nos expériences. Pour certaines d'entre elles, nous n'avons pas affiché tous les graphiques. Cependant nous avons mis à disposition un dépôt Github [1] contenant notre code source ainsi que la totalité des graphiques. Ici, nous avons affiché l'évolution de la population pour quelques-uns des réseaux de zombies. Pour chacune, nous affichons sa courbe d'infection minimale, maximale, moyenne et médiane de l'ensemble des simulations de l'expérience.

4.3.1 EXPÉRIENCE 1A

Cette première expérience a pour but d'observer la concurrence entre divers réplicats d'un même réseau de zombies. Ici, nous considérons un cas où tous les réplicats commencent à chercher des victimes en même temps. Notre objectif est de trouver le temps à partir duquel le système devient stable et que les populations n'évoluent plus de manière significative. Ce cas de figure représente la phase primaire d'infection, nous ne montrons pas la phase de vie, où les divers réseaux vont grappiller les nouveaux hôtes ou ceux ayant subi un redémarrage provoquant la disparition de l'infection.

La Figure 4.3 représente l'évolution du premier ver (appelé ici Mirai0, car adoptant la même stratégie de scan) dans la configuration où cinq réplicats se disputent les victimes. La Figure **??** représente l'évolution du troisième réplicat (appelé ici Mirai2) dans la configuration où cinq réplicats se disputent les victimes.

1. `https://github.com/bvignau/The_Botnet_Game`

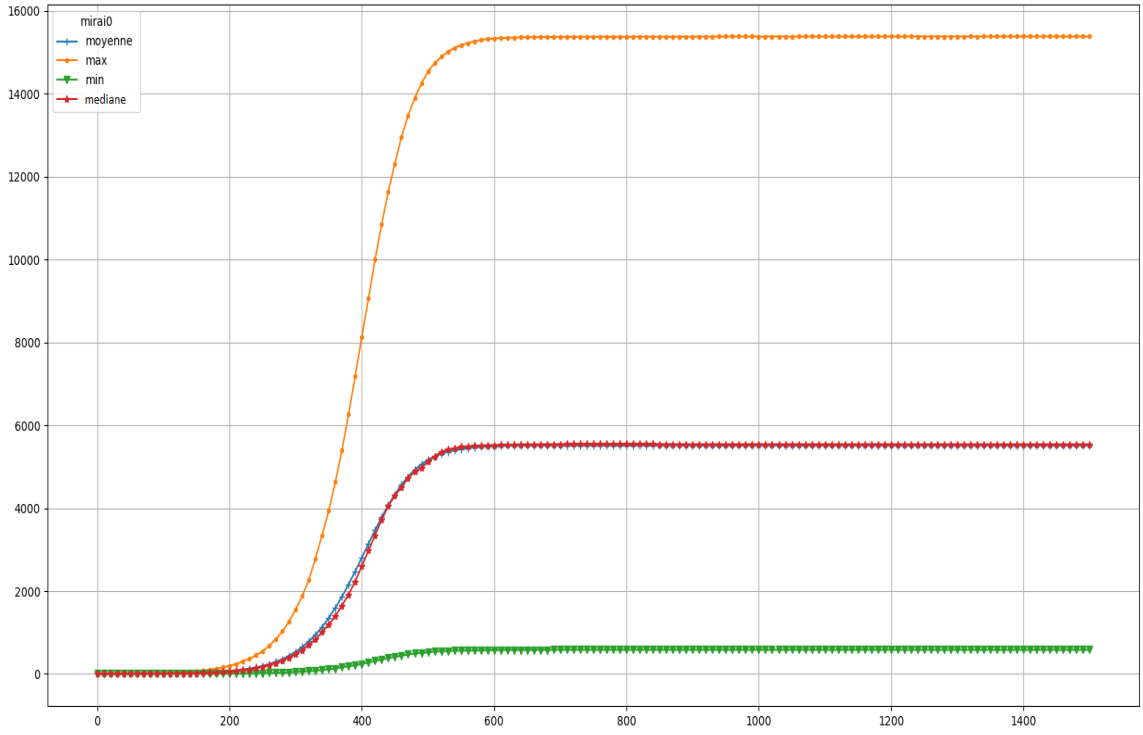

Figure 4.3 – Évolution de la population du réseau #1 sur 1 500 tours (configuration avec 5 vers)

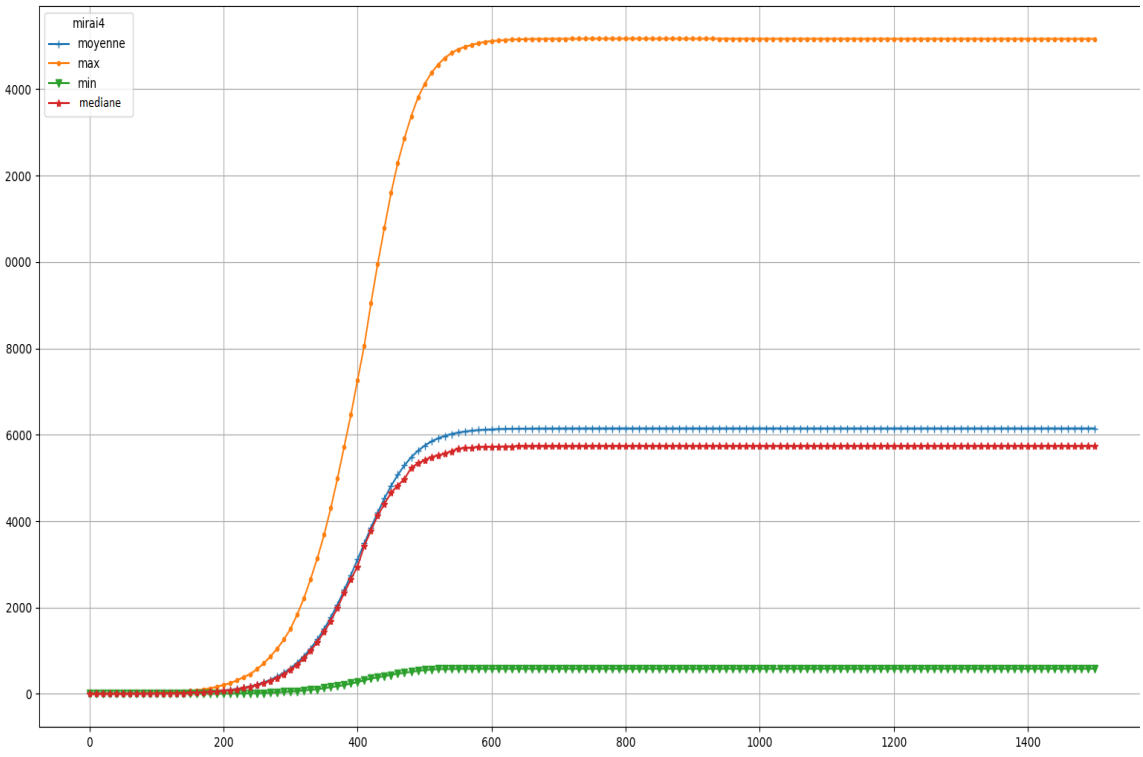

Figure 4.4 – Évolution de la population du botnet #5 sur 1500 tours (configuration avec 5 vers)

La Figure 4.4 représente l'évolution du cinquième réplicat (appelé ici Mirai4) dans la configuration où cinq réplicats se disputent les victimes. Ici, nous ne montrons qu'un ver sur deux car les résultats sont très similaires. Cependant, l'ensemble des figures des populations de chaque réplicat est disponible sur Github.

Pour cette expérience, on observe une grande disparité entre les populations de zombies. Les différences entre maximaux et minimaux sont importantes. On peut observer sur les graphiques représentant les populations individuelles (Figure 4.3 **??**), que leur minimum est d'environ 750 individus et leur maximum à plus de 15 000. Cette grande variation s'explique par le côté aléatoire de la stratégie de scan. On observe donc qu'une technique de scan aléatoire possède une efficacité très variable en milieu compétitif.

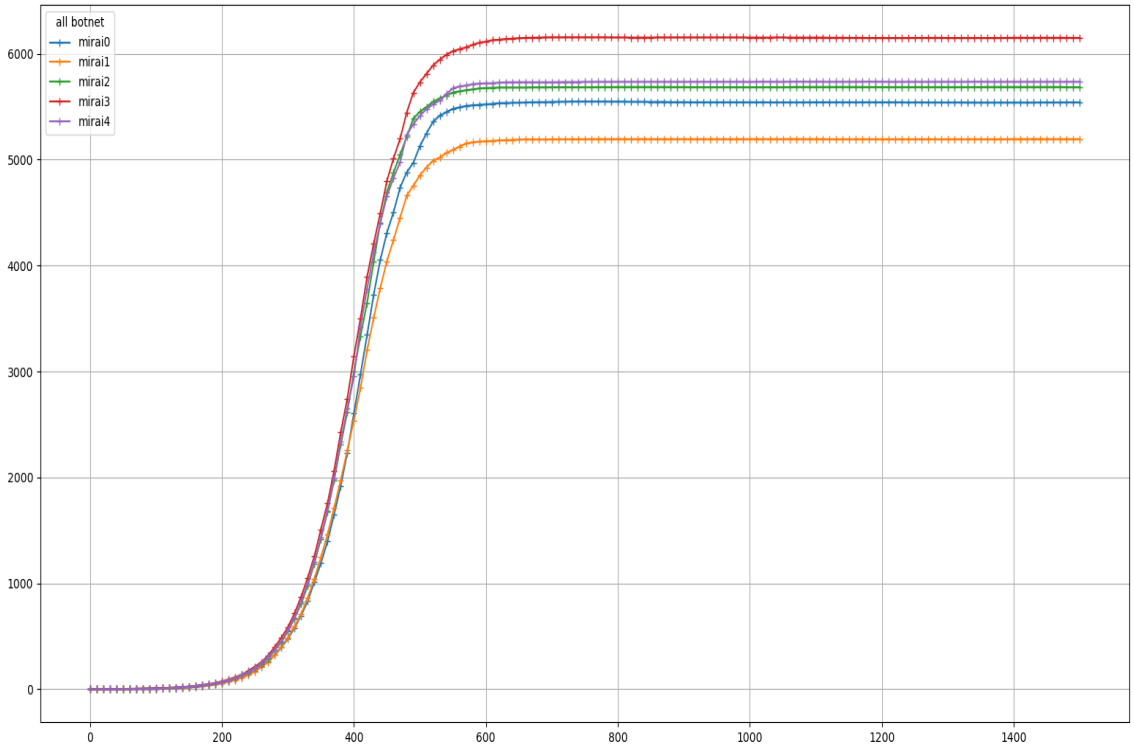

Figure 4.5 – Évolution de la population des 5 botnets sur 1500 tours (configuration avec 5 vers)

Sur la figure 4.5 nous affichons les courbes d'infections médianes de l'ensemble des réseaux de

zombies. Nous pouvons observer que ces dernières sont plutôt homogènes. L'écart de population entre la plus faible et la plus forte est aux environs de 1 000 individus. De plus, comme tous les réseaux vont attaquer la même population, on observe une division des populations de chaque réseau. Chaque réseau possède entre 16% et 21% du total victimes.

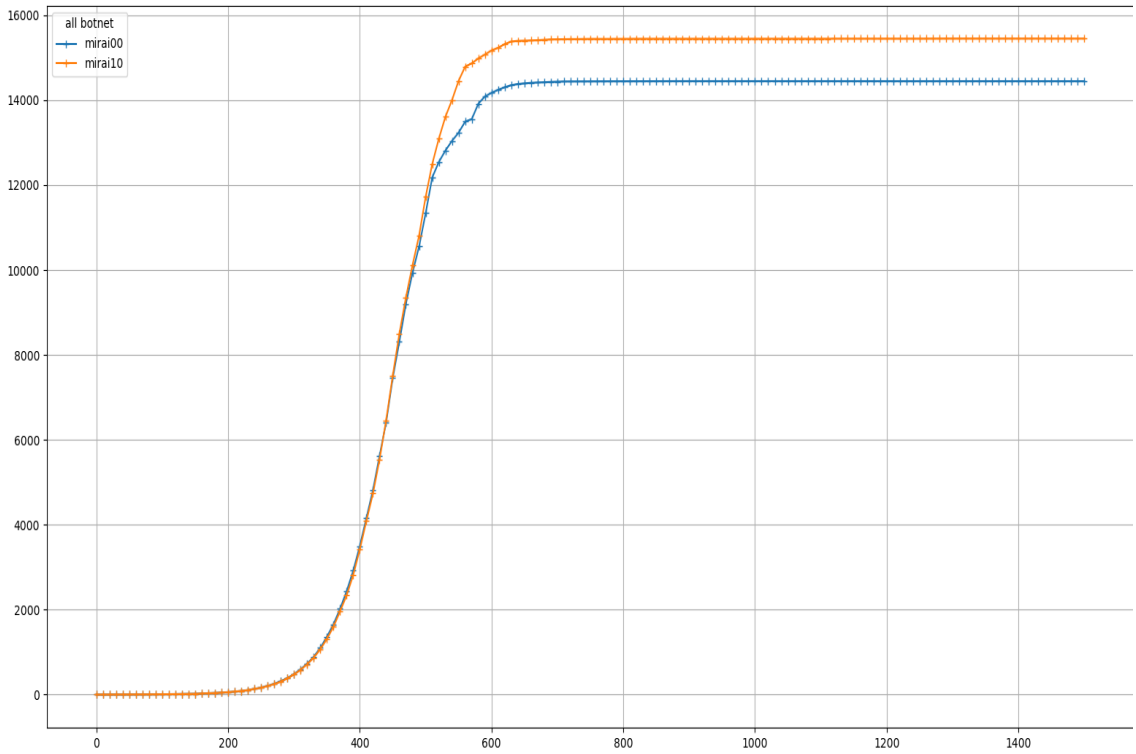

Figure 4.6 – Évolution de la population des 2 botnets sur 1500 tours (configuration avec 2 vers)

Sur la figure 4.6 nous observons les populations médianes des réseaux de zombies dans une configuration à deux réseaux. Ici, nous observons que les deux populations médianes sont proches et se retrouvent aux alentours des 15 000 individus. Cela correspond à environ 50% de la totalité de la population vulnérable.

On observe donc que plus il y a de réseaux, plus la puissance de chaque réseau sera diminuée, de manière quasi proportionnelle aux nombres de compétiteurs. Ainsi, une première solution pour réduire la puissance d'un réseau de zombies serait de lancer plusieurs réseaux du même type

dans un temps proche. Cependant, cette solution requiert une analyse et une modification très rapide d'un programme malveillant. En pratique, il y aura forcément un délai entre le moment où le programme originel commencera son attaque et le moment où une équipe pourra déployer un concurrent à ce réseau.

4.3.2 EXPÉRIENCE 1B

Cette seconde version de la première expérience vise à évaluer l'impact d'un retard au départ dans la capacité d'infection d'un réseau de zombies. Ainsi, nous avons lancé les mêmes réseaux de zombies, mais avec des temps de départs différents. Cela peux représenter, par exemple, le fait qu'une personne créé un réseaux de zombie avec un code, et que plus tard, une autre personne utilise le même code pour créer une instance différente et concurrente du réseaux de zombie initial.

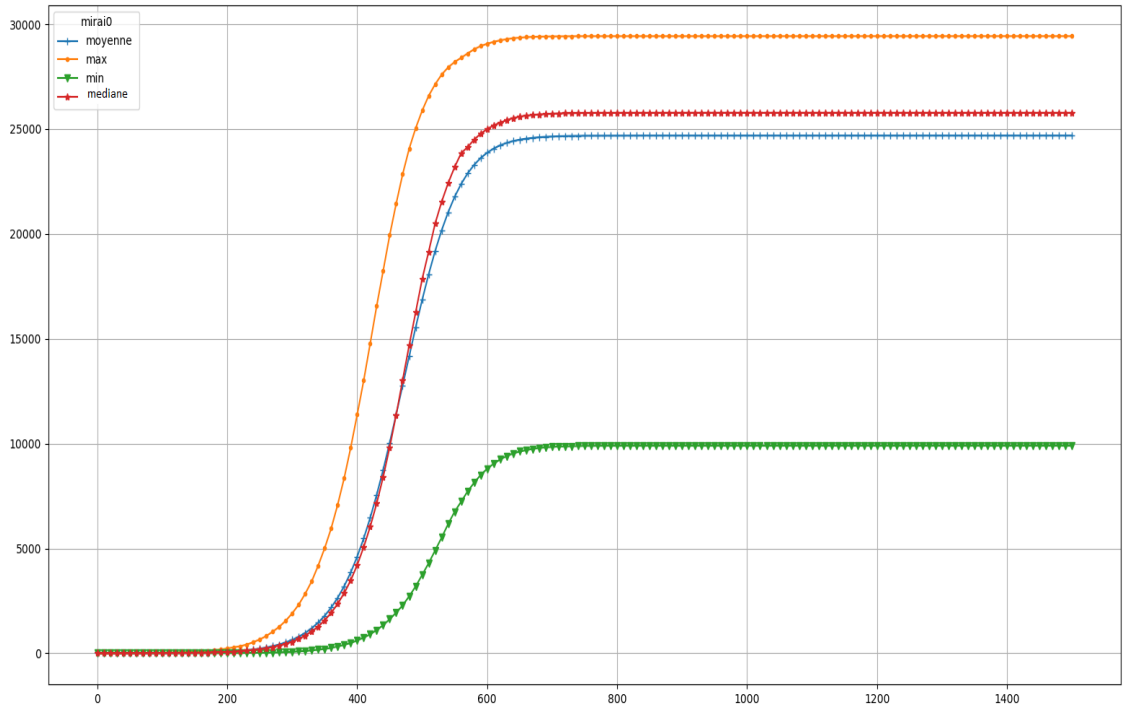

Figure 4.7 – Évolution de la population du botnet #1 sur 1500 tours

Sur la figure 4.7, nous pouvons observer l'évolution de la population du premier réseau de zombie. On constate toujours un grand écart entre le maximum et le minimum mais ce dernier est bien moins grand que pour l'expérience précédente. Ici, le minimum est aux alentours de 10000 individus et le maximum vers 30000, soit la totalité de la population vulnérable. Cependant, on observe que la médiane se situe un peut au dessus des 25000 individus, montrant ainsi une efficacité très importante dans une moitié des cas. Comme pour la première expérience, nous ne montrons que les résultats d'un réseau sur deux. L'ensemble des résultats est sur notre Github.

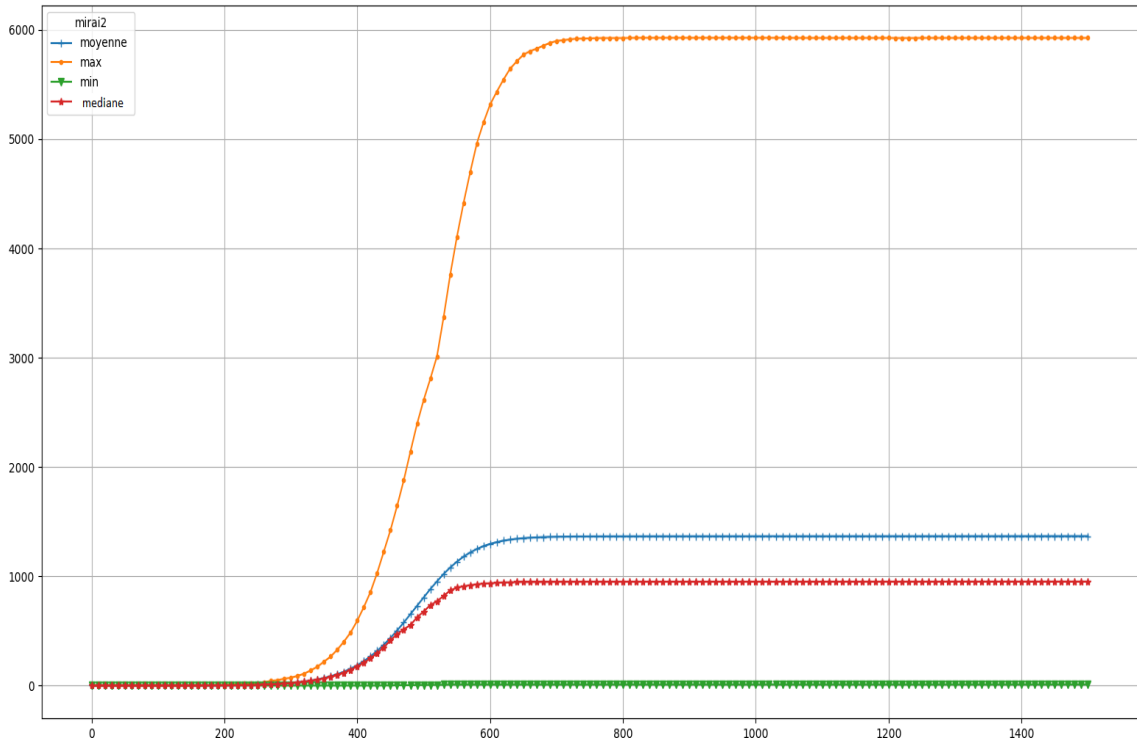

Figure 4.8 – Évolution de la population du botnet #3 sur 1500 tours

Sur la figure 4.8 nous observons l'évolution du troisième réseau de zombies, parti avec 150 tours de retard. Ici aussi, l'écart entre le maximum et le minimum est très élevé, environ 6 000 individus au maximum et presque 200 au minimum. Encore une fois, cela montre la grande variabilité de l'efficacité de cette technique. De plus, contrairement au réseau précédent, la population médiane et la population moyenne sont faibles, aux alentours de 1 000 individus.

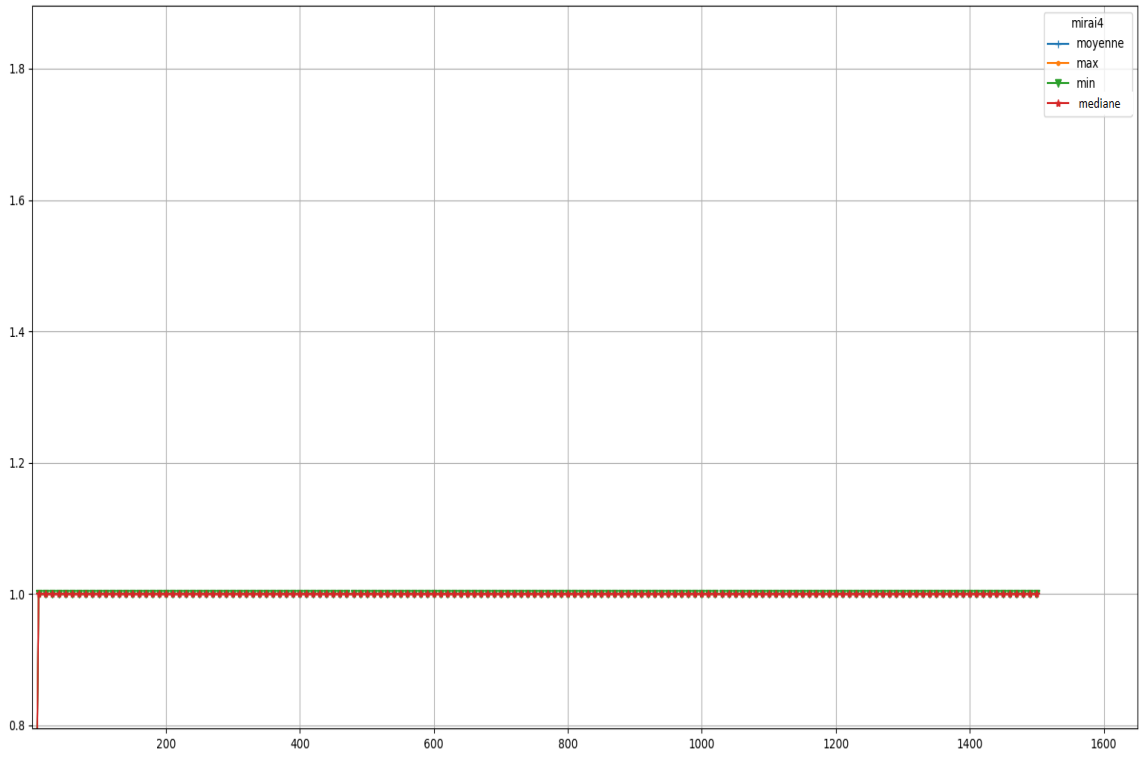

Figure 4.9 – Évolution de la population du botnet #5 sur 1500 tours

Pour le dernier réseau de zombies, démarré avec 1000 tours de retard, il n'a pas été capable d'infecter une seule victime. En effet, on observe sur l'expérience précédente que le point d'équilibre est atteint entre 600 et 800 tours. Or ici, en partant après ce point d'équilibre, le réseau ne peut pas infecter de nouvelles victimes.

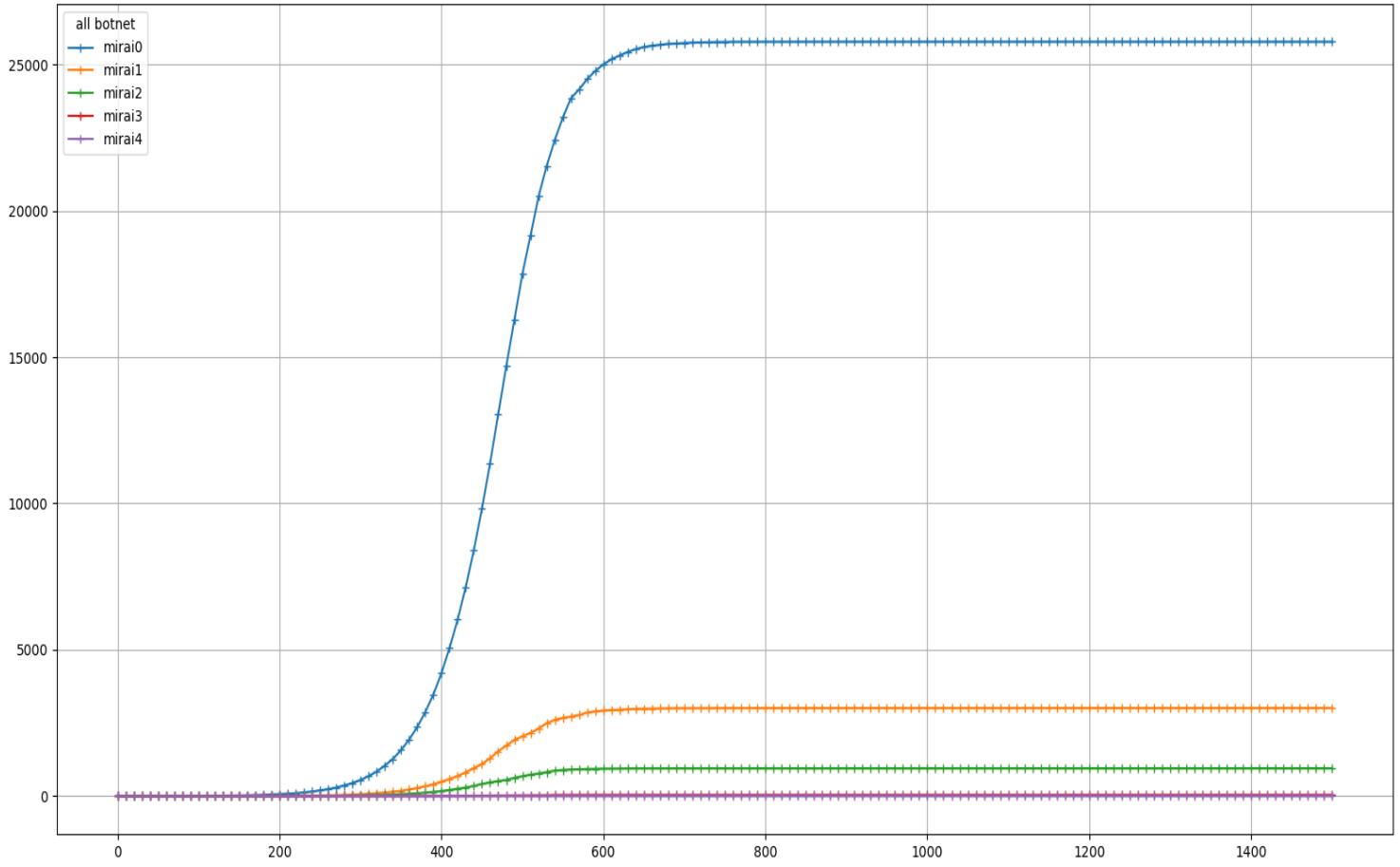

Figure 4.10 – Évolution de la population des 5 botnets sur 1500 tours

Finalement, en observant les populations médianes de chaque réseau, on observe que l'efficacité de chaque réseau décroît avec le temps de retard. Ceci est parfaitement logique étant donné que chaque réseau va infecter des victimes que les autres réseaux ne pourront récupérer. De plus, cette stratégie d'infection possède un point à partir duquel l'efficacité est exponentielle. Ainsi, avoir un départ retardé influence énormément le nombre de zombies que peut recruter un réseau. 100 tours de retards entraînent une grande différence entre les réseaux de zombies.

Le premier obtient plus de 25 000 individus représentant ici plus de 80% de la population vulnérable. Le second réseau n'en obtient qu'environ 4 000. Ainsi, on observe que par rapport à la première expérience où tous les réseaux démarrent simultanément, le premier réseau de zombies a fortement gagné en efficacité, là où les autres ont perdu significativement.

4.3.3 EXPÉRIENCE 2A

Pour cette expérience, nous avons souhaité observer la compétition entre deux modes d'infection. Le premier est le scan séquentiel et le second le scan aléatoire comme utilisés précédemment. Ici, nous supposons que le scan séquentiel sera moins performant que le scan aléatoire du fait que seul le premier zombie pourra ajouter de nouvelles recrues. Ici aussi, notre but est d'observer l'évolution des systèmes jusqu'à leur point d'équilibre. Nous avons fait les expériences sur 1000 et 5000 tours.

Les figures 4.11 et 4.12 représentent le réseau 1 (appelé ici Psybot), utilisant le scan séquentiel. La première figure représente l'évolution de la population sur 1000 tours et la seconde sur 5000tours. Les figures 4.13 et 4.14 représentent le réseau 2 (appelé ici Mirai), utilisant le scan aléatoire. La première figure représente l'évolution de la population sur 1 000 tours et la seconde sur 5 000 tours.

On peut d'ores et déjà observer que le point d'équilibre du système est atteint vers 800 tours et qu'à partir de là le système n'évolue plus. Dans cette expérience, on observe que le réseau 1 démontre une faible efficacité, variant entre une dizaine et une vingtaine d'individus. Cette variation s'explique par la distribution aléatoire des victimes, changeant d'une simulation à une autre. À l'opposé, le réseau deux réussis à accaparer la majorité de la population. Cependant, contrairement à la première expérience, on observe que la population totale termine toujours aux alentours de 30000 individus. La différence entre chaque simulation se fait au niveau du

temps d'apparition du point d'équilibre. Ainsi, dans le meilleur des cas, on observe aux figures 4.13 et 4.14 un point d'équilibre peut après le tour 600. Ce point arrive autour du tour 800 dans le pire des cas.

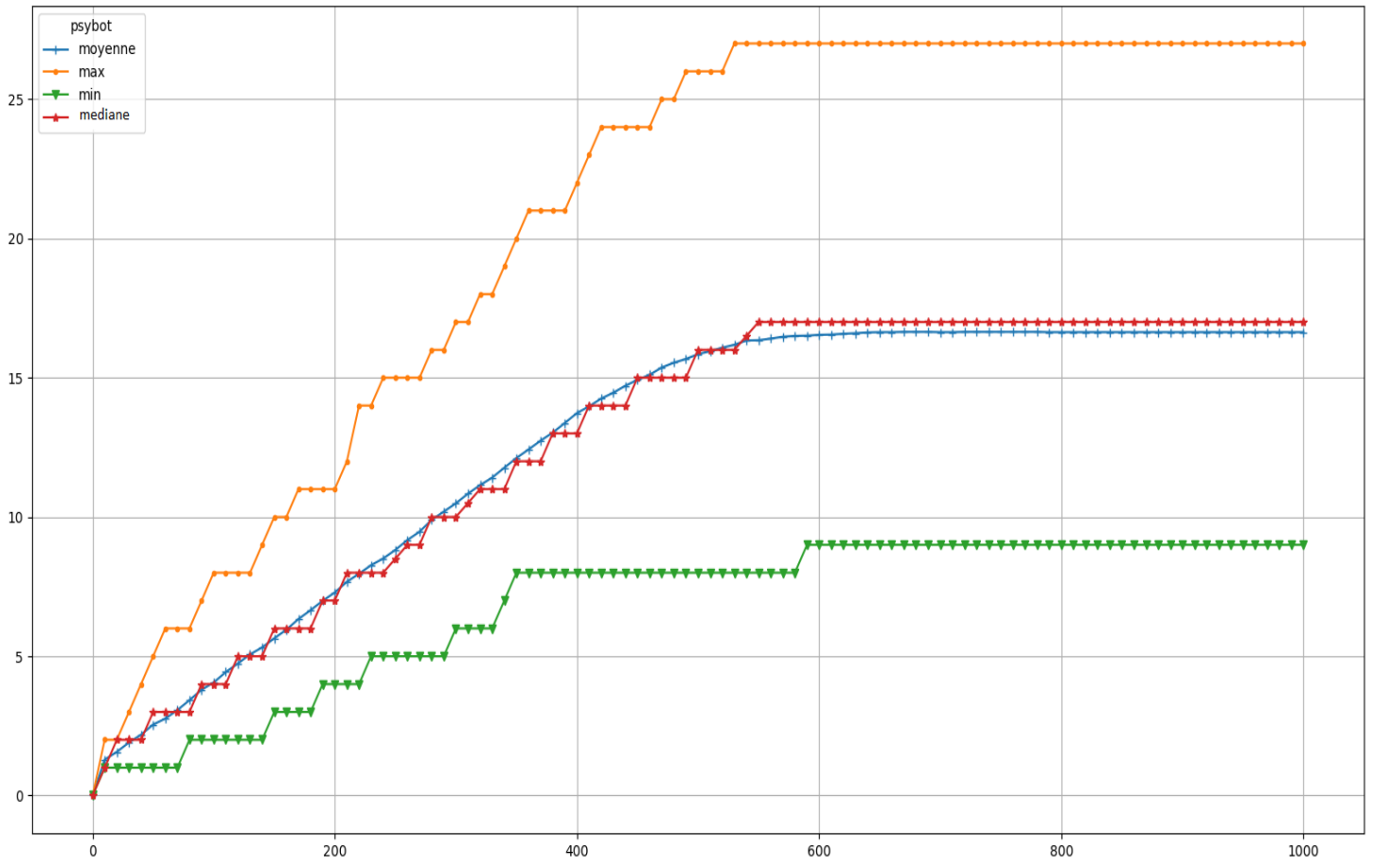

Figure 4.11 – Évolution de la population du botnet #1 sur 1000 tours

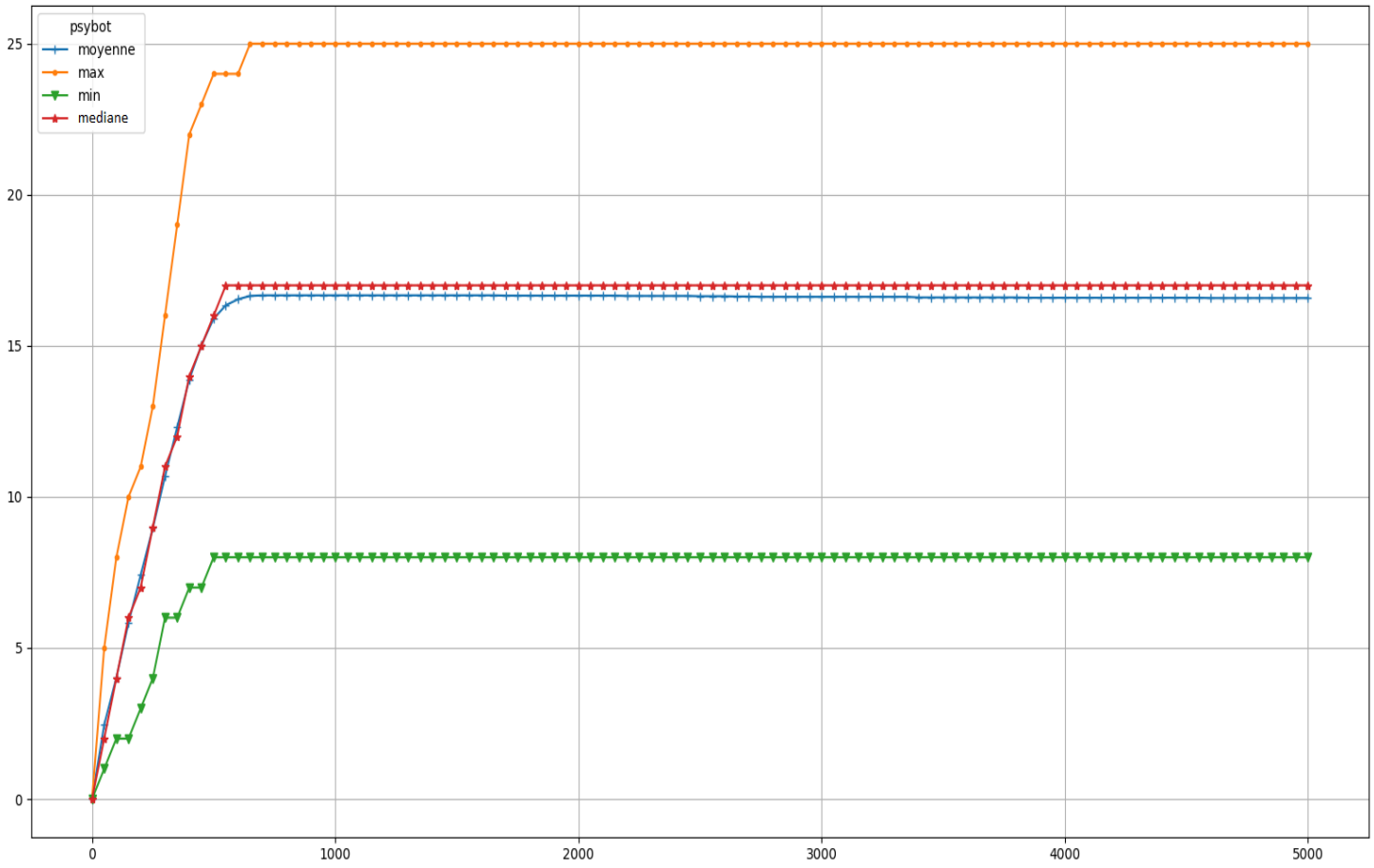

Figure 4.12 – Évolution de la population du botnet #1 sur 5 000 tours

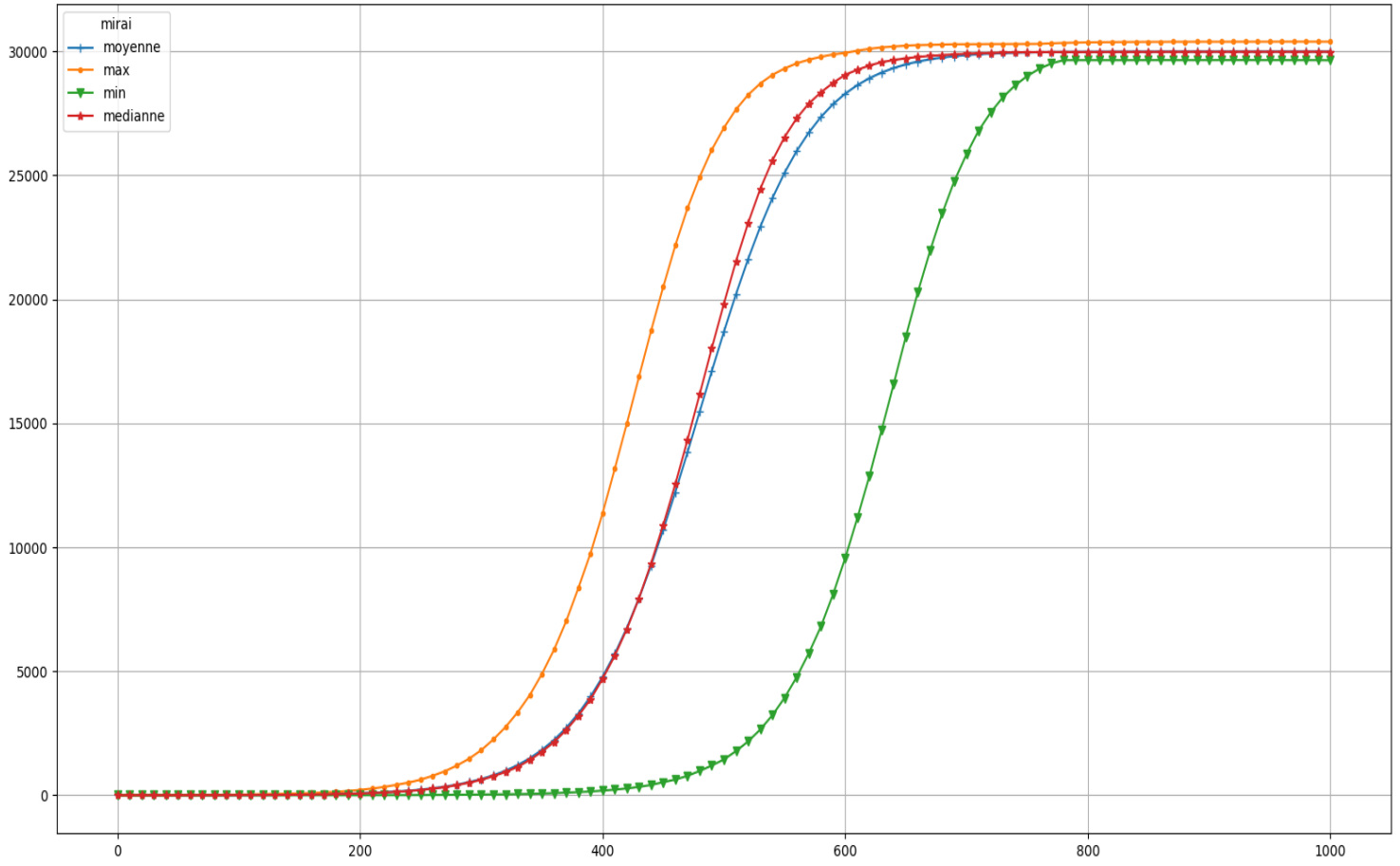

Figure 4.13 – Évolution de la population du botnet #2 sur 1 000 tours

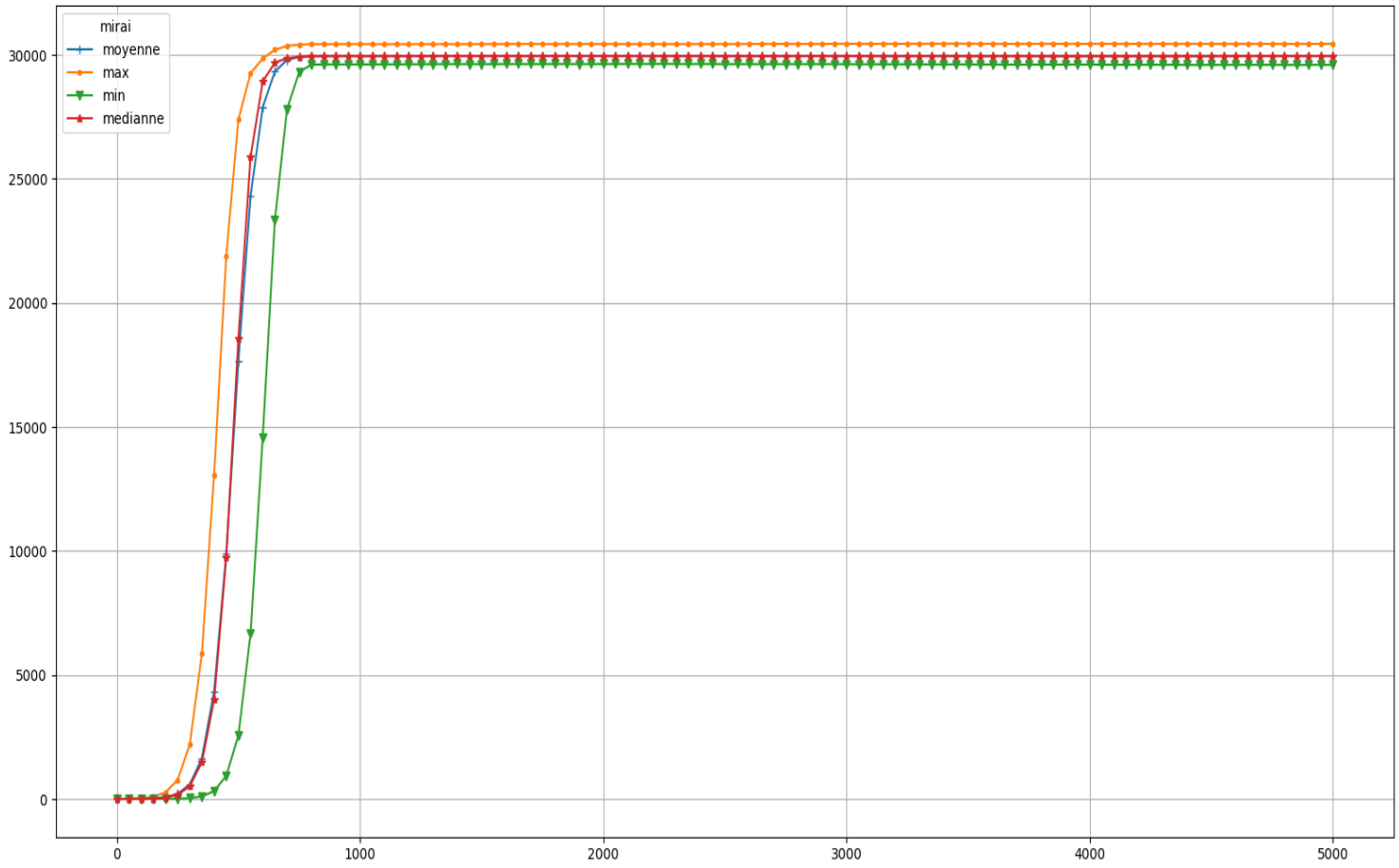

Figure 4.14 – Évolution de la population du botnet #2 sur 5 000 tours

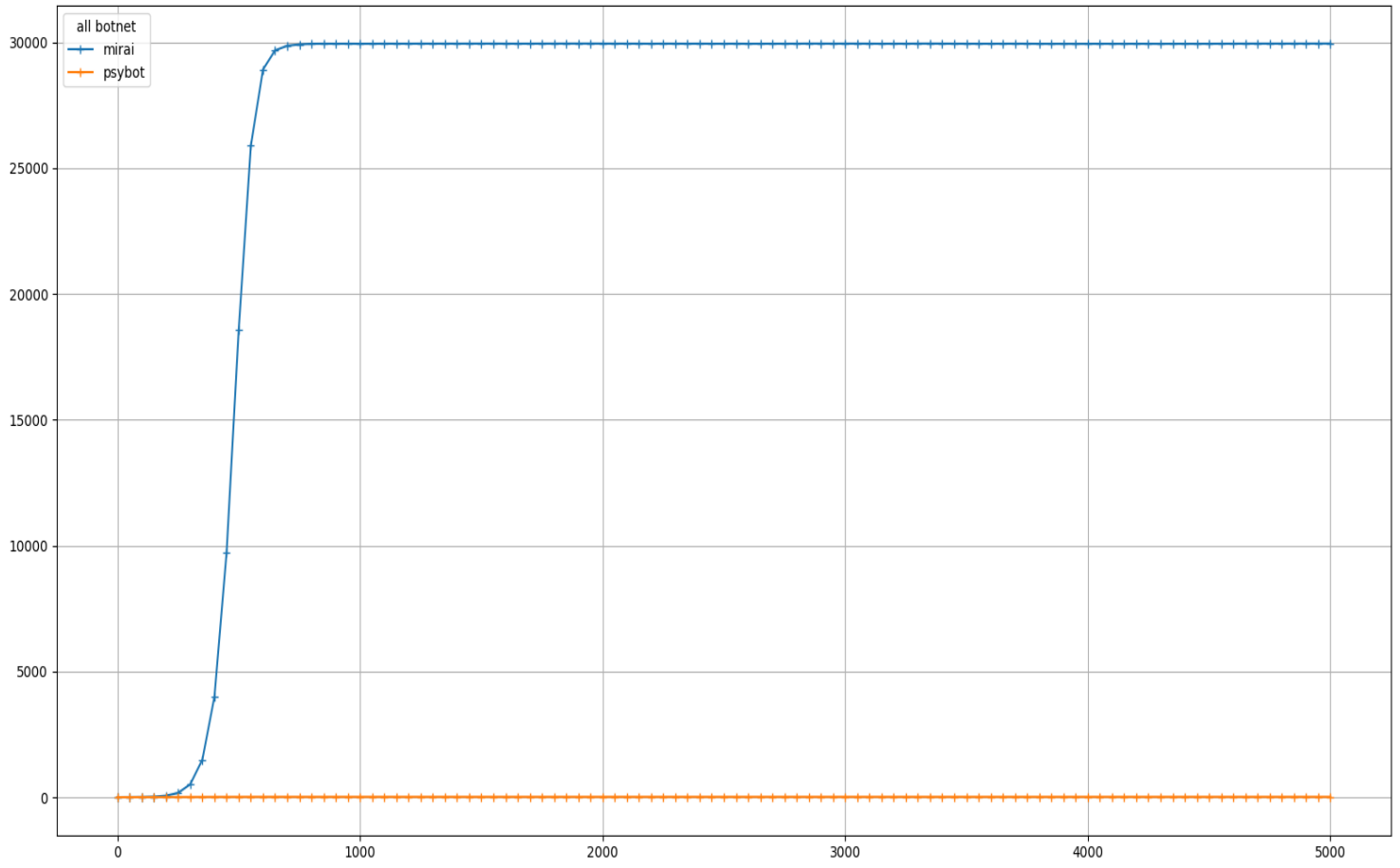

Figure 4.15 – Évolution de la population des deux botnet sur 5 000 tours

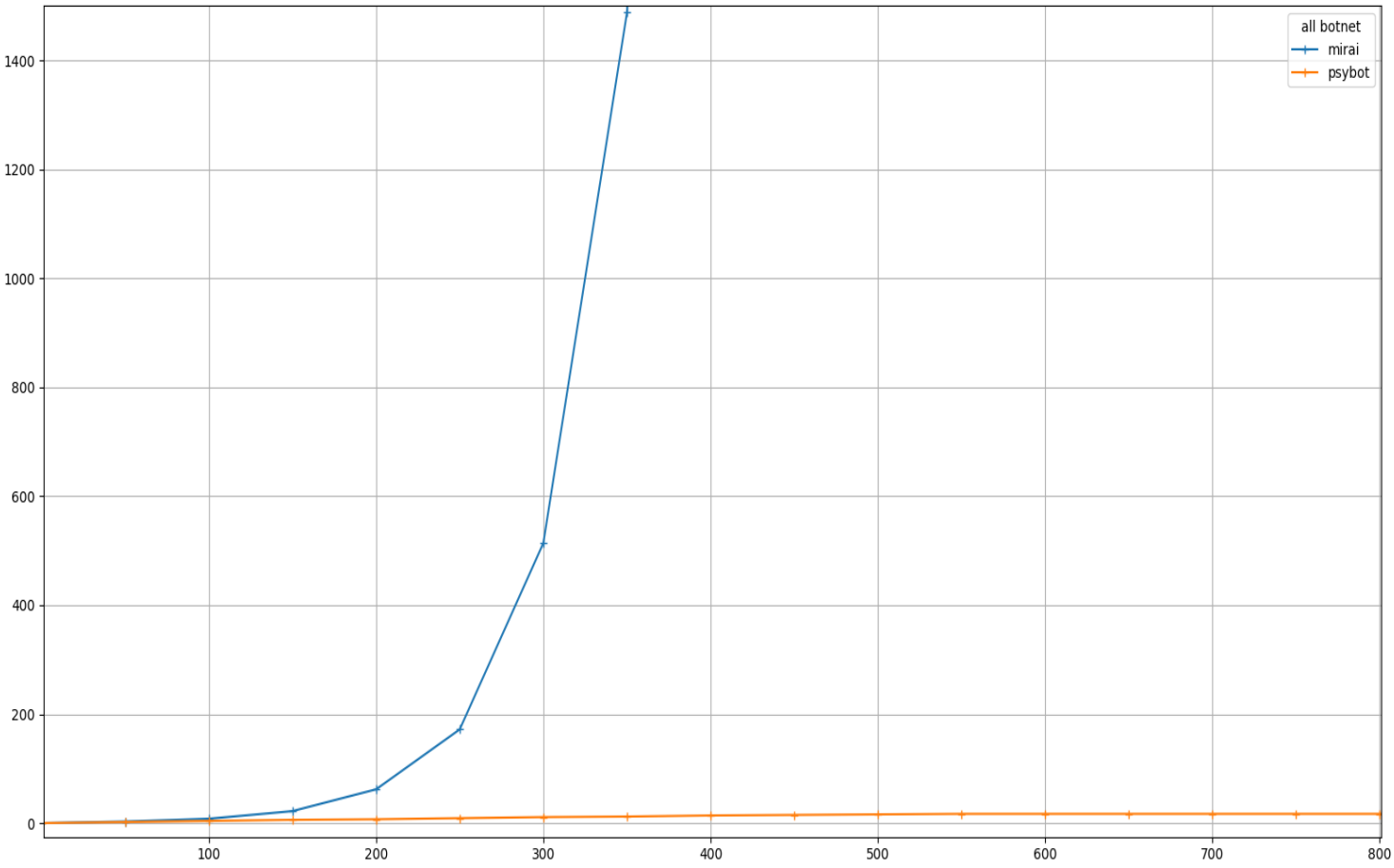

Figure 4.16 – Zoom de l'évolution de la population des deux botnet sur 5 000 tours

Les figures 4.15 et 4.16 représentent l'évolution de la population médiane des deux réseaux. On y observe que le réseau 2 est beaucoup plus efficace que le réseau 1 et le surpasse largement à partir de 150 tours. En effet, l'évolution du réseau utilisant le scan séquentiel est très lente et régulière, alors que le scan aléatoire possède une évolution exponentielle à partir d'un certain nombre d'appareils infectés. Ici, ce point arrive entre 150 et 200 tours.

4.3.4 EXPÉRIENCE 2B

Pour cette expérience, nous avons modifié les temps que prend chaque action pour le réseau 1, qui utilise le scan séquentiel. En effet, nous souhaitions observer l'impact du temps de processus sur cette stratégie et voir si cela permettait d'améliorer significativement son efficacité. Ainsi, nous avons divisé par 5 et par 4 les temps des phases de scans et d'infections. À noter que le temps de génération d'une adresse IP était 3 fois moins important pour le réseau 1 que pour le second.

Les figures 4.17 et 4.18 représentent le réseau 1 (appelé ici Psybot), utilisant le scan séquentiel. La première figure représente l'évolution de la population sur 1 000 tours et la seconde sur 5 000 tours. La figure 4.19 est un agrandissement de la figure 4.18. Les figures 4.20 et 4.21 représentent le réseau 2 (appelé ici Mirai), utilisant le scan aléatoire. La première figure représente l'évolution de la population sur 1 000 tours et la seconde sur 5 000 tours. La figure 4.22 est un agrandissement de la figure 4.21. Ici, les résultats sont proches de l'expérience 2A. L'efficacité du réseau 1 est meilleure que dans l'expérience précédente, mais reste extrêmement faible par rapport au réseau 2 utilisant le scan aléatoire. Son efficacité a plus que doublé, en passant en moyenne de 16 à 35 individus. Cela reste cependant négligeable par rapport aux 30 000 individus vulnérables.

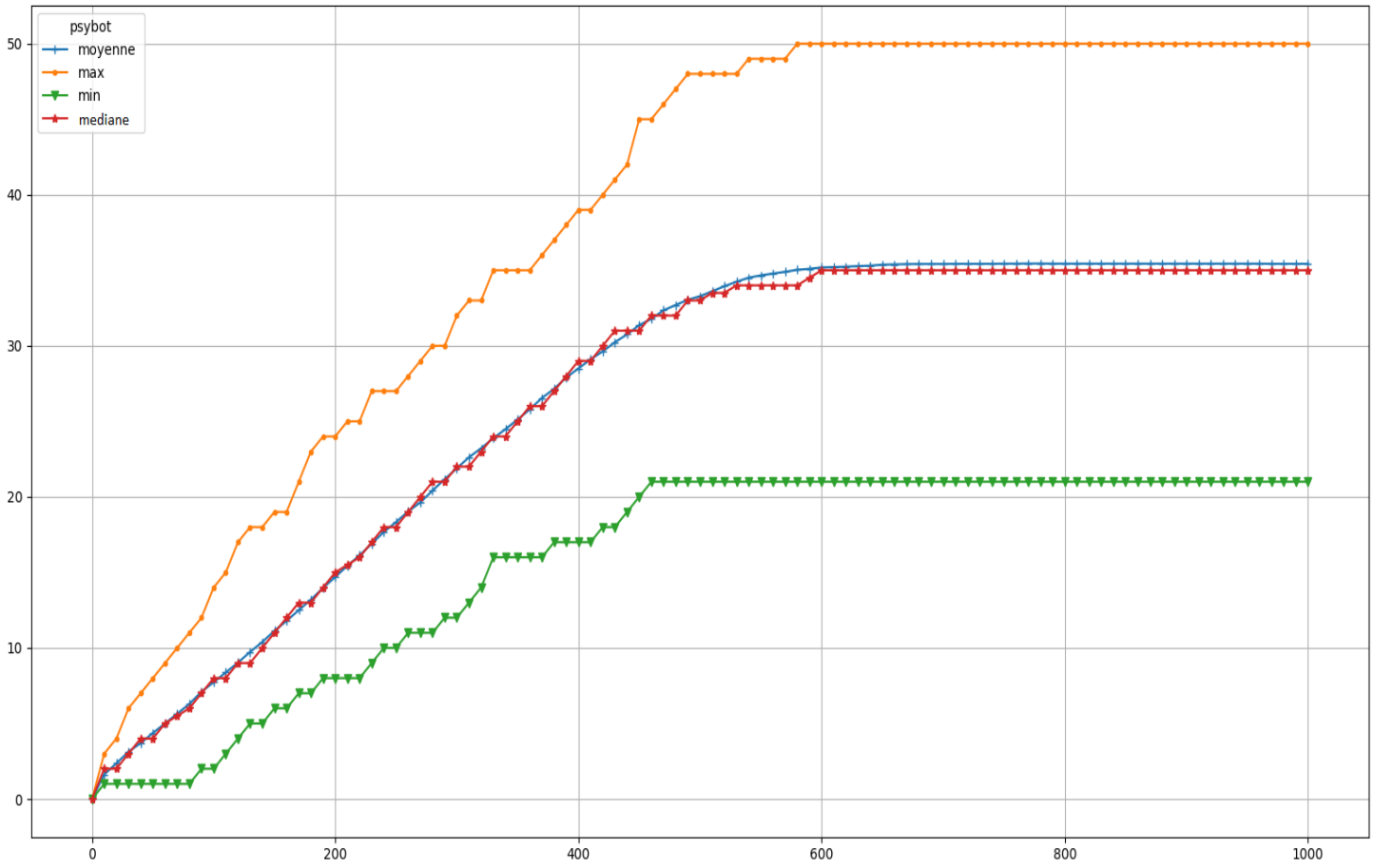

Figure 4.17 – Évolution de la population du botnet #1 sur 1000 tours

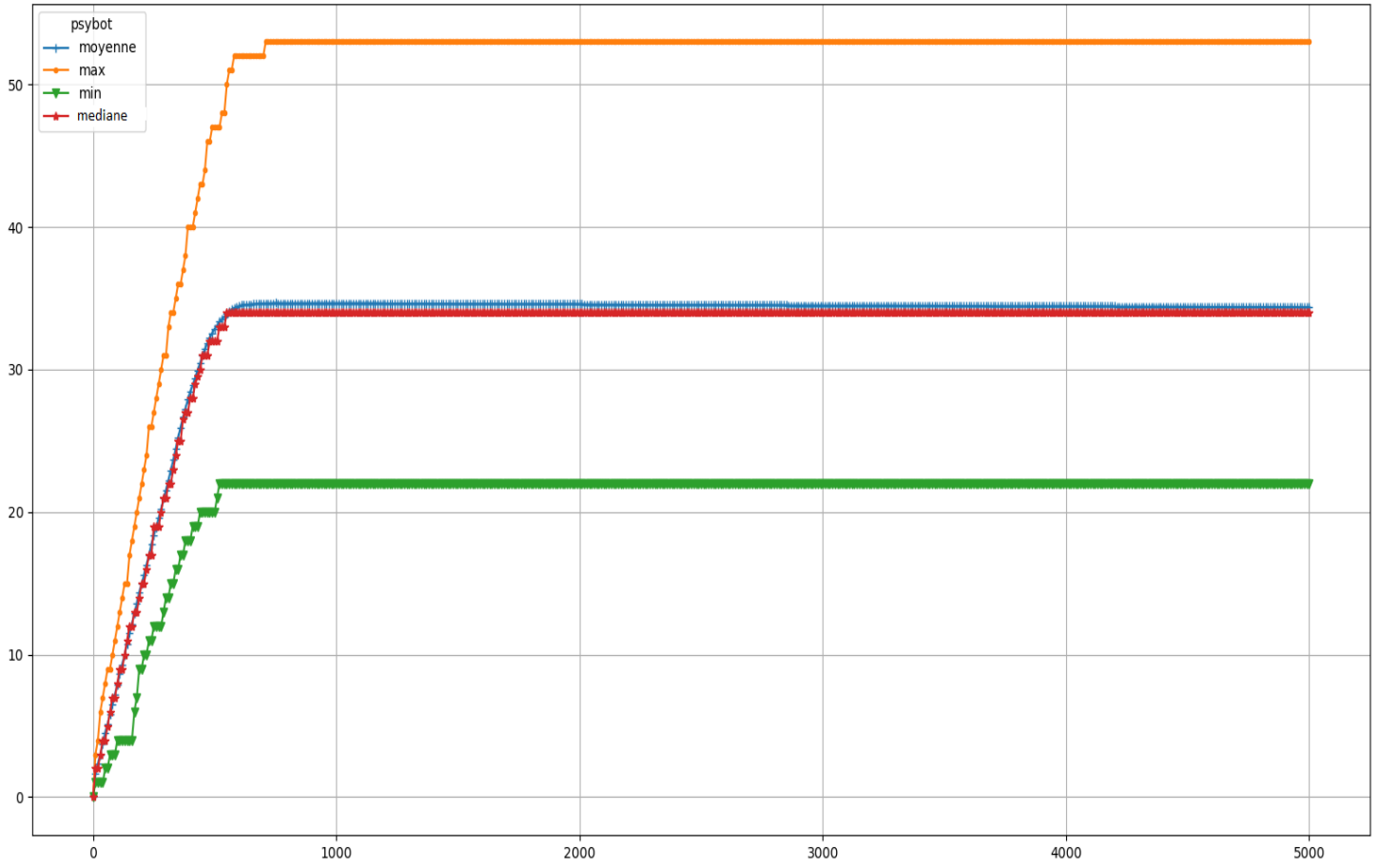

Figure 4.18 – Évolution de la population du botnet #1 sur 5000 tours

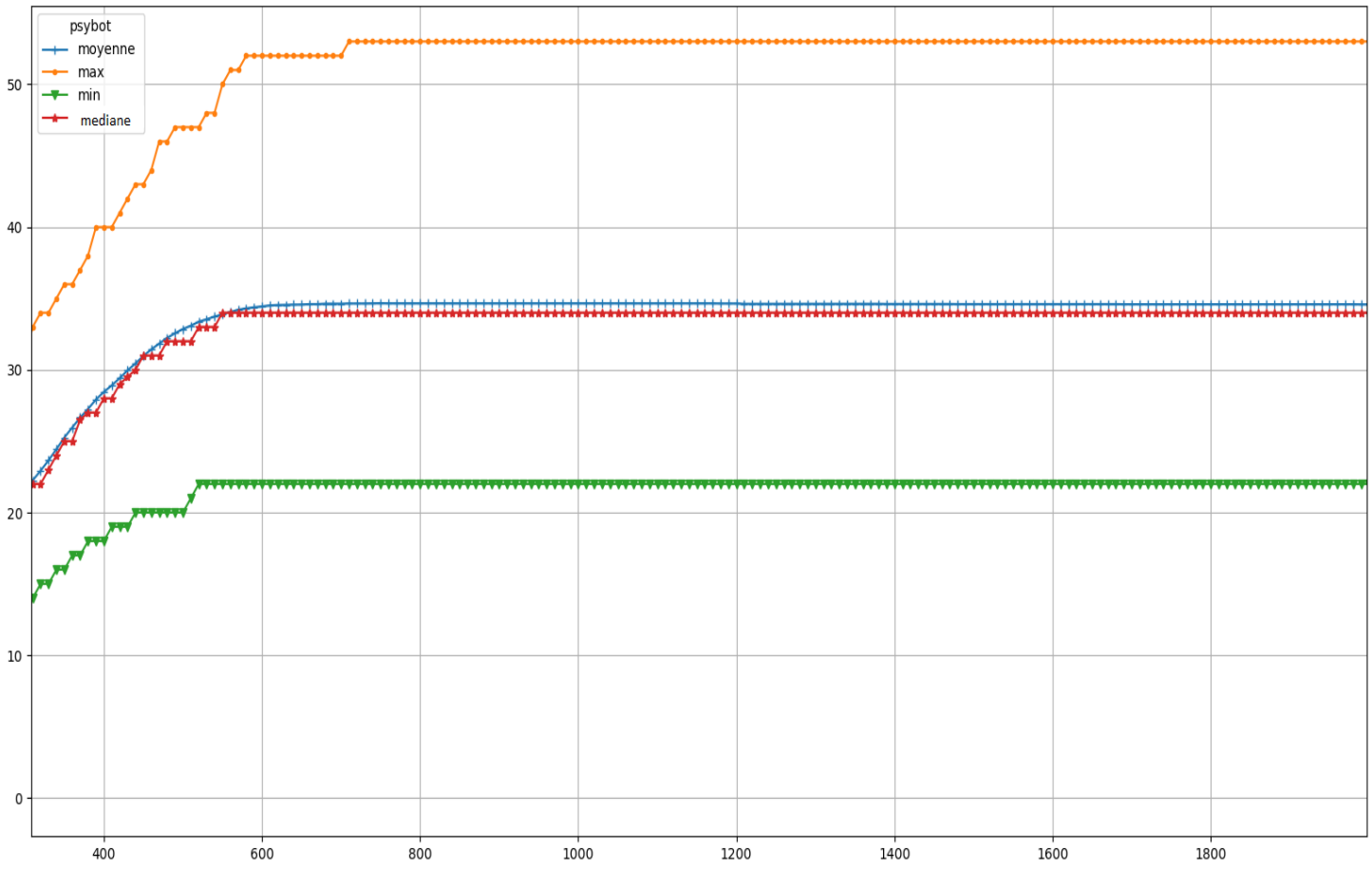

Figure 4.19 – Zoom de l'évolution de la population du botnet #1 sur 5000 tours

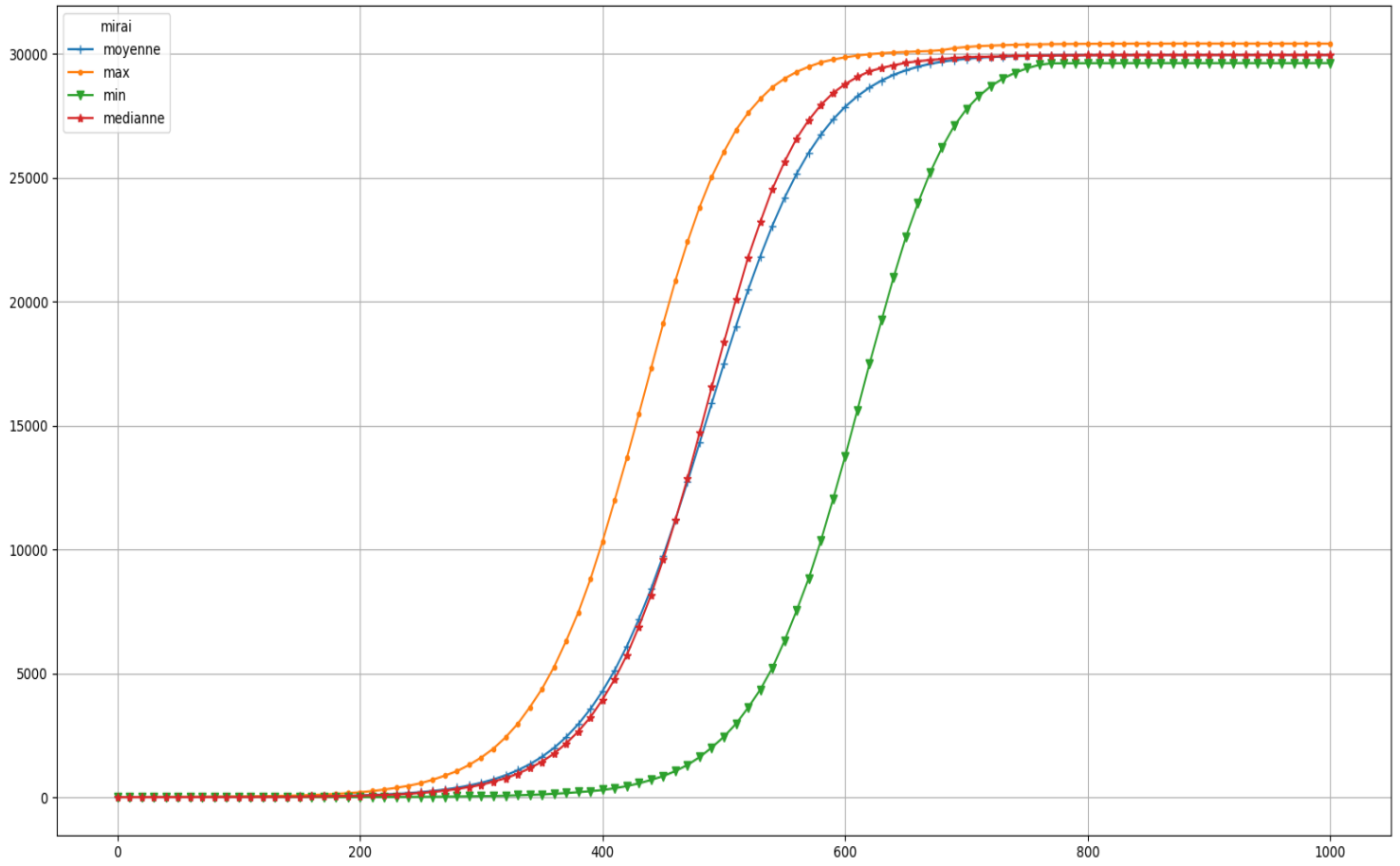

Figure 4.20 – Évolution de la population du botnet #2 sur 1000 tours

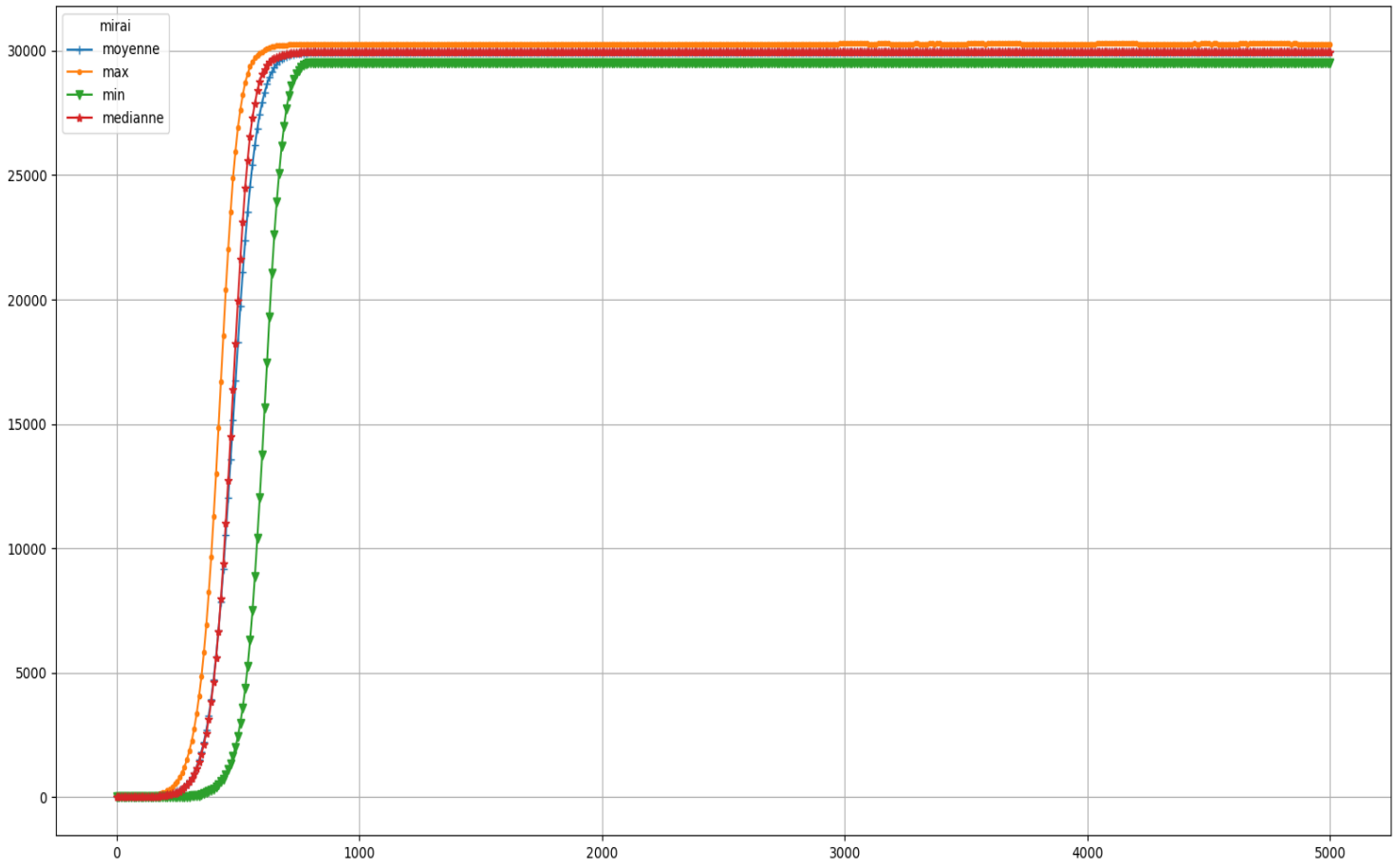

Figure 4.21 – Évolution de la population du botnet #2 sur 5000 tours

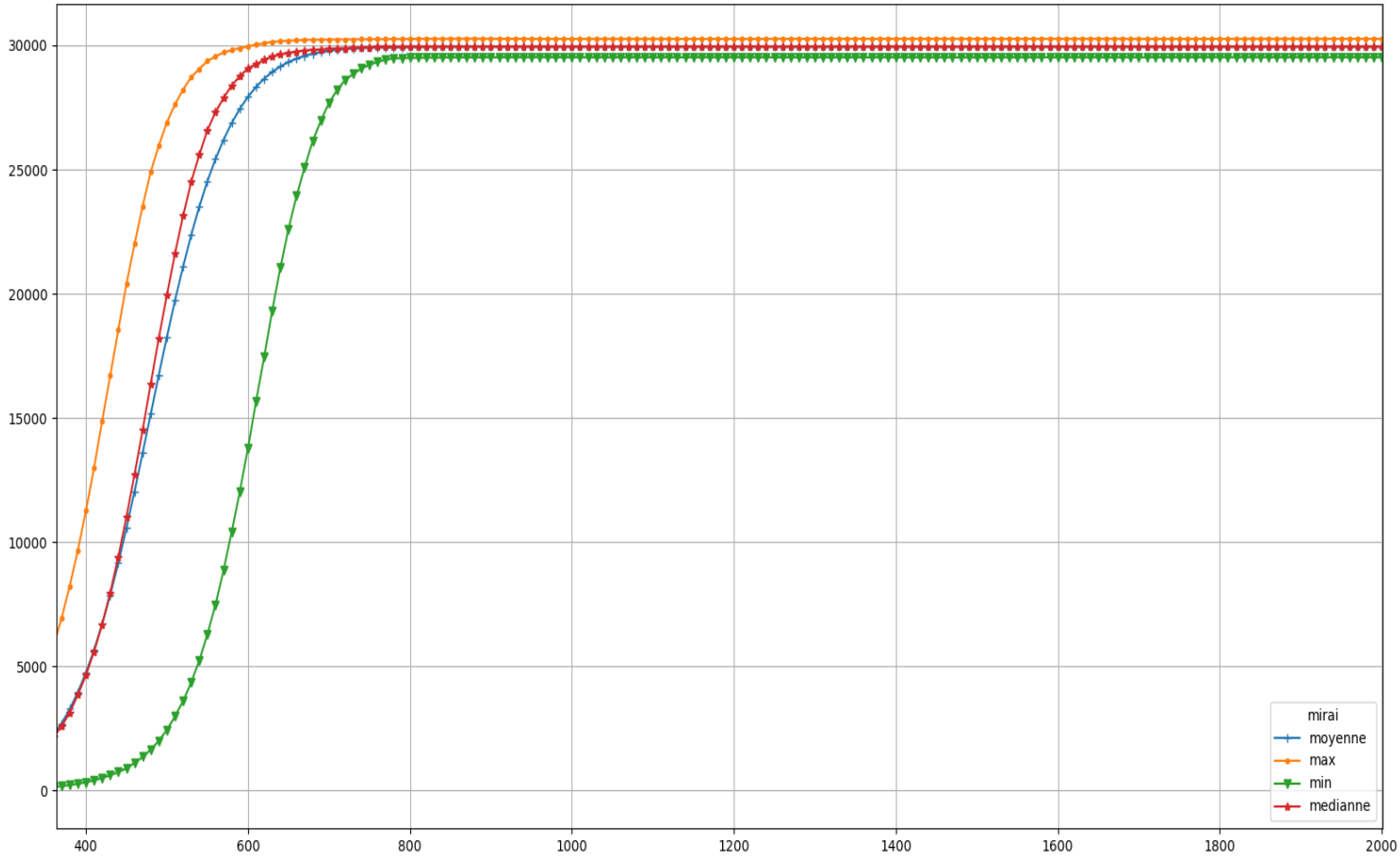

Figure 4.22 – Zoom de l'évolution de la population du botnet #2 sur 5000 tours

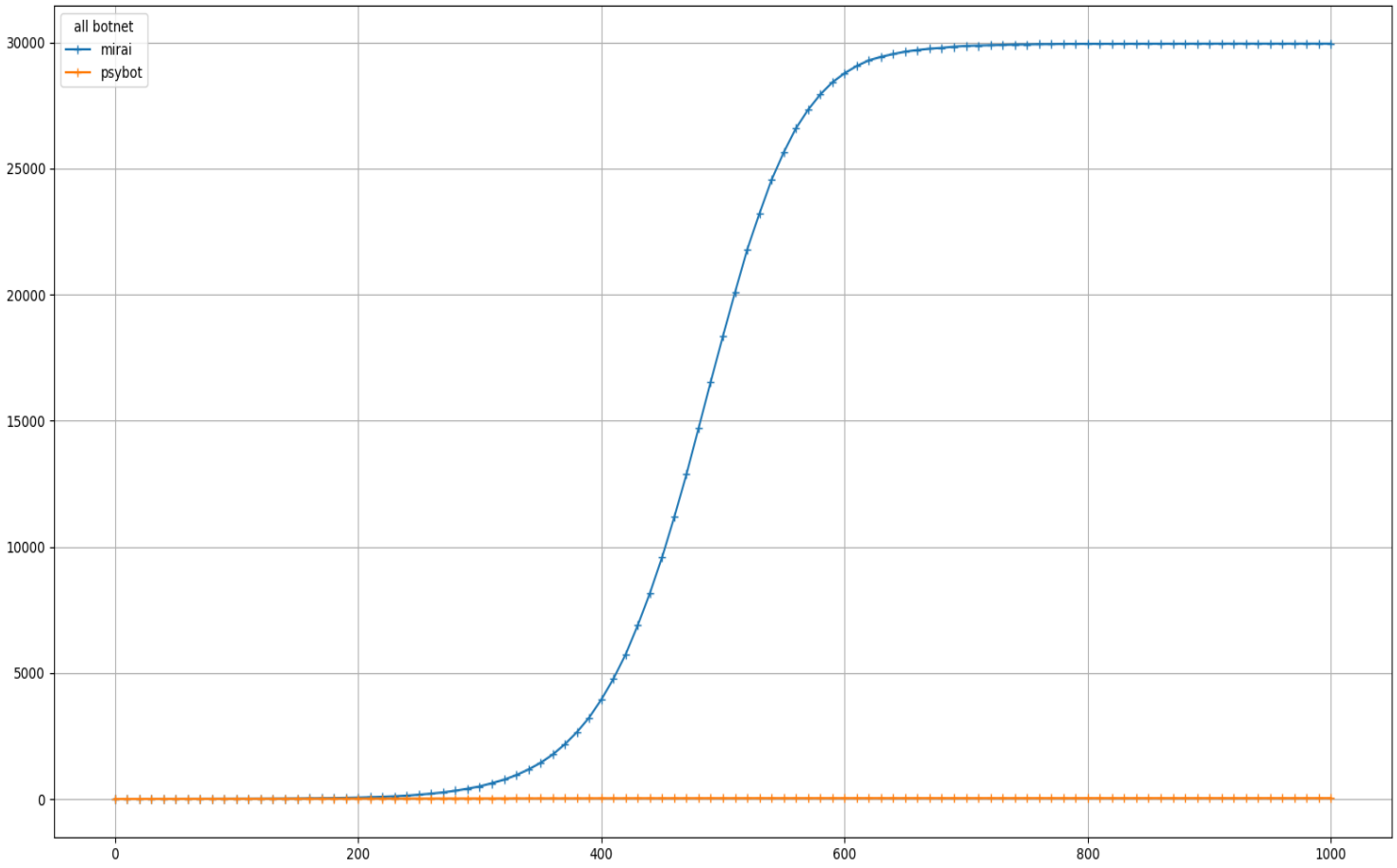

Figure 4.23 – Évolution de la population des deux botnets sur 1000 tours

4.3.5 EXPÉRIENCE 3A

Pour cette expérience, nous souhaitions observer l'impact d'un ver capable d'infecter les objets déjà infectés par d'autres programme malveillants et de supprimer l'infection, tout en immunisant la victime. Ce cas de figure est apparu ces dernières années. En effet, comme le précise Radaware dans un de ses articles de blogs (Daniel, 2018), la libération du code source

de Mirai a entraîné une saturation de réseaux de zombies. Ainsi, chaque réseau a dû innover, en incluant des systèmes de correctif pour supprimer les autres réseaux de zombies. Ils ont aussi commencé à exploiter différentes failles afin d'avoir de nouvelles méthodes pour recruter des zombies.

Pour cette première version, nous avons utilisé deux réseaux de zombies identiques, utilisant le scan aléatoire et le second immunisant ses victimes contre les deux réseaux. Le second est aussi capable de supprimer l'infection du premier réseau. De plus, le second réseau part avec un retard de deux cents tours, soit suffisamment pour que le premier réseau puisse atteindre une croissance exponentielle.

La figure 4.24 représente l'évolution de la population du premier réseau de zombies, n'immunisant pas ses victimes. La figure 4.25 représente l'évolution de la population du second réseau de zombies, capable d'immuniser ses victimes et de les désinfecter du premier réseau. La figure 4.26 représente l'évolution de la population médiane des deux réseaux.

On peut ainsi observer sur la figure 4.25 que la population du premier réseau est capable de grandir rapidement, pouvant parfois infecter la quasi-totalité de la population vulnérable. Cependant, aux alentours du tour 600, la population chute très rapidement, pour atteindre 0 individu entre le tour 800 et 1 000. Au contraire, la population du second réseau augmente normalement et finit par atteindre l'équilibre vers le tour 1 000, soit avec 200 tours de retard par rapport au cas classique.

Sur la figure 4.26 on observe que le point d'inversion de courbe du premier réseau, correspond au point d'inflexion de la courbe du second réseau. Ainsi, au moment où le second réseau entame sa phase de croissance exponentielle, il va « voler » des victimes au premier réseau.

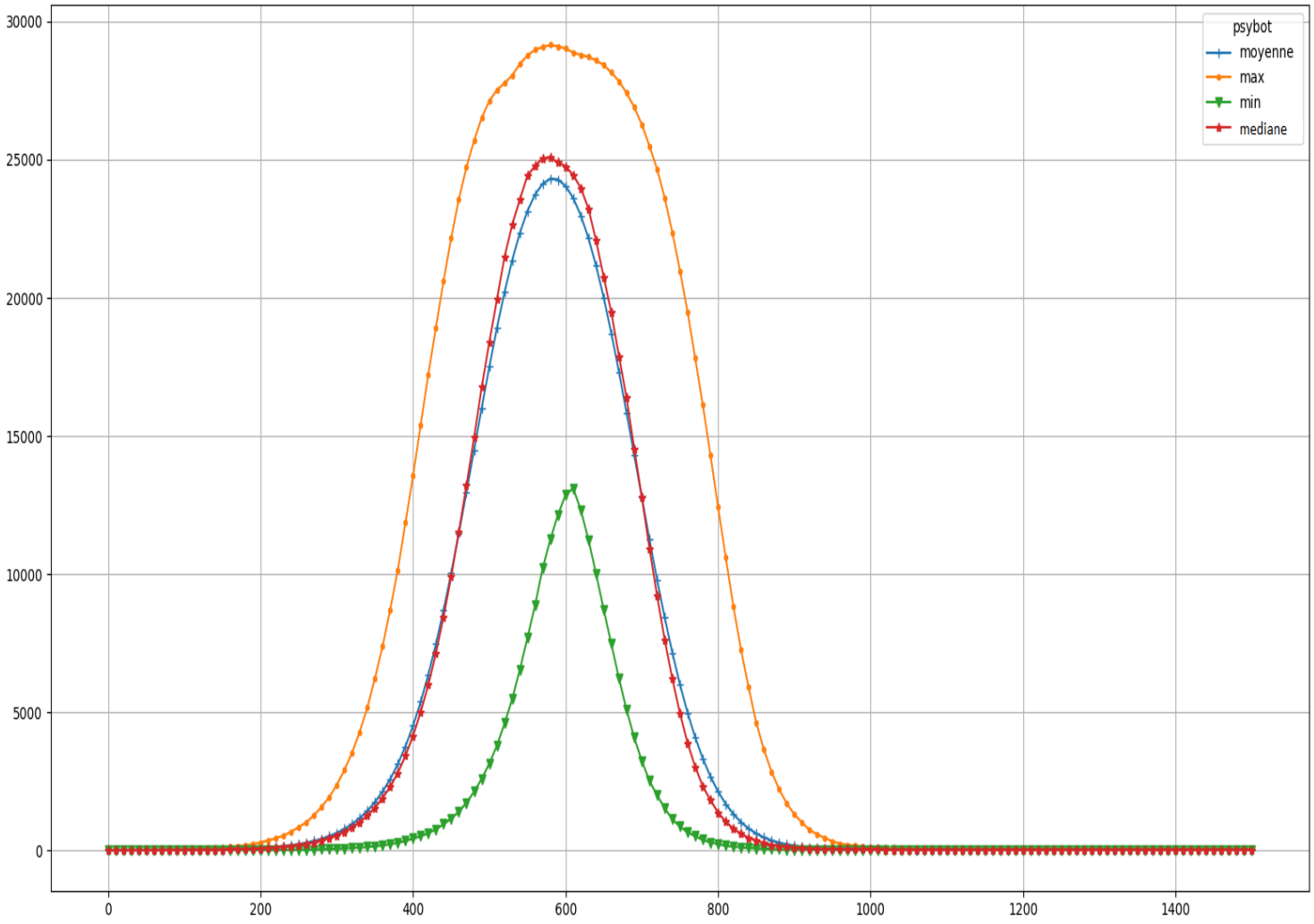

Figure 4.24 – Évolution de la population du botnet #1 sur 1 500 tours

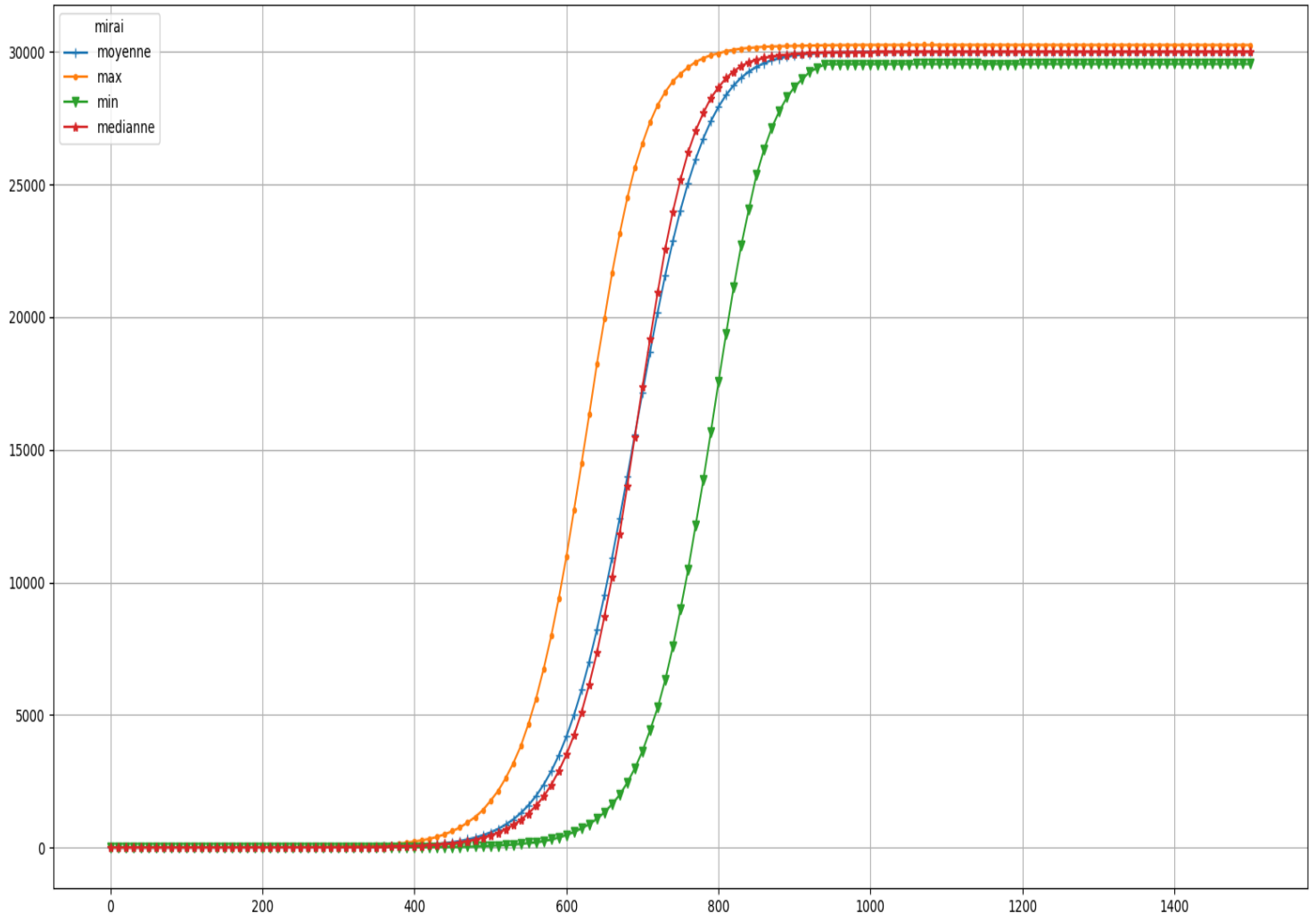

Figure 4.25 – Évolution de la population du botnet #2 sur 1 500 tours

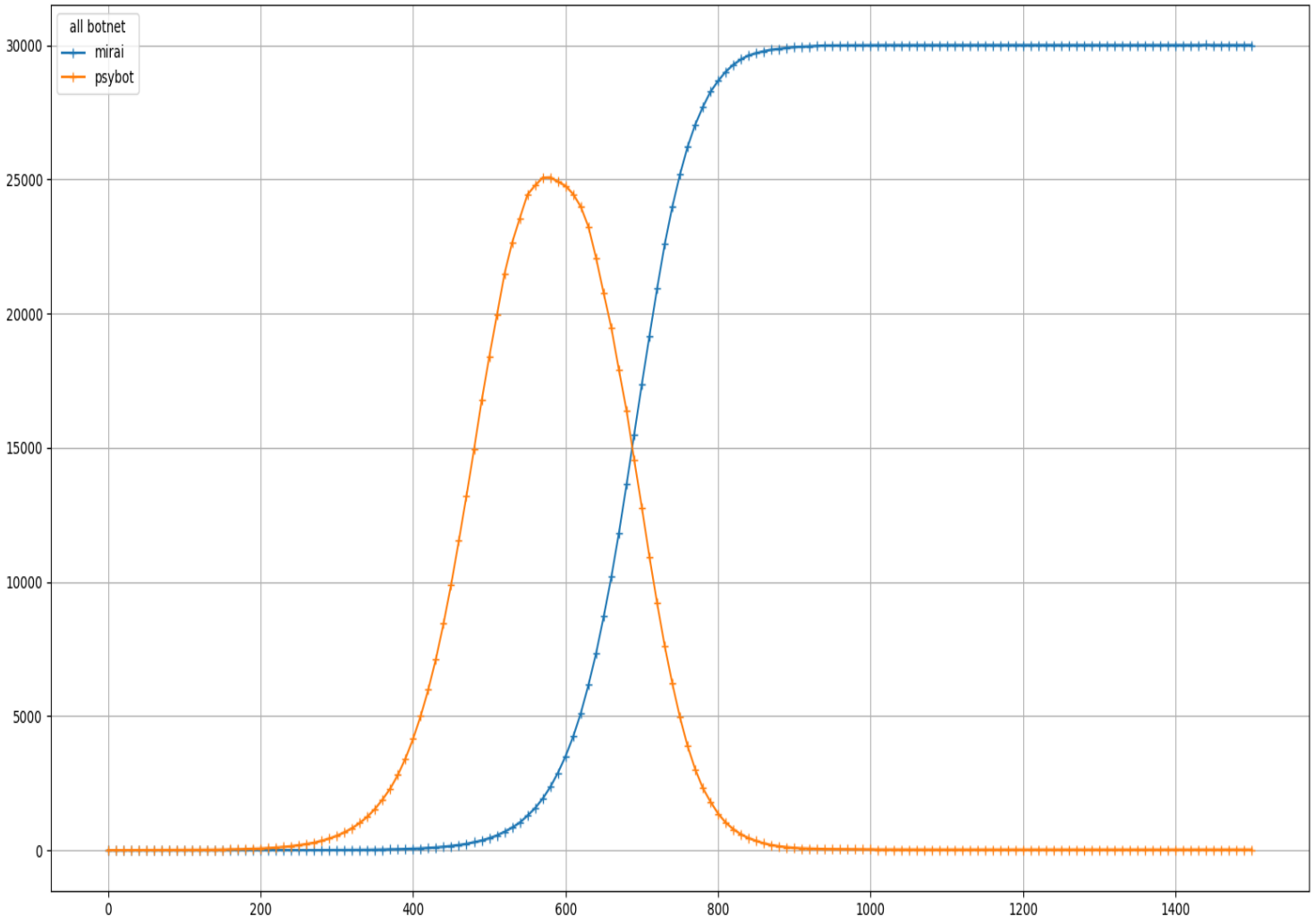

Figure 4.26 – Évolution de la population des deux botnet sur 1 500 tours

4.3.6 EXPÉRIENCE 3B

Cette expérience est la même que la précédente, à la différence que le premier réseau utilise un scan séquentiel tel que défini plus tôt. La figure 4.27 représente l'évolution de la population du

premier réseau. La figure 4.28 représente l'évolution de la population du second réseau. Enfin, la figure 4.29 représente l'évolution des populations médianes des deux réseaux. Les résultats sont similaires à la première expérience, à la différence que le premier réseau atteint au maximum 55 individus et non 30 000. L'évolution du second réseau est ici identique à son homologue de l'expérience 3A. A la figure 4.29 on observe bien que le premier réseau ne parvient à recruter que peu victimes et que ses dernières finissent par se faire recruté par le second réseau. A la fin on observe bien la mort du premier réseau terminant avec 1 seul individus, le serveur originel.

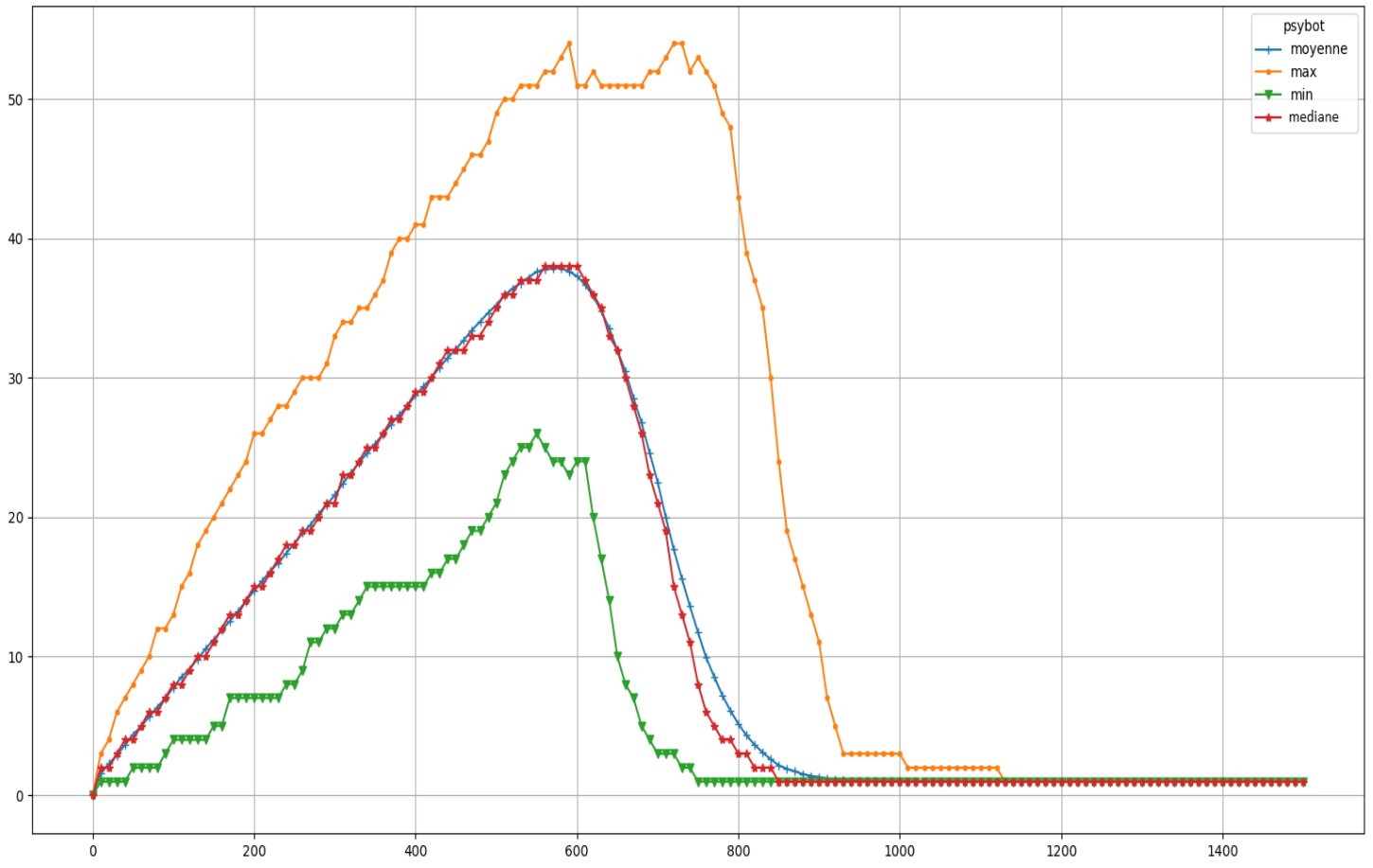

Figure 4.27 – Évolution de la population du botnet #1 sur 1500 tours

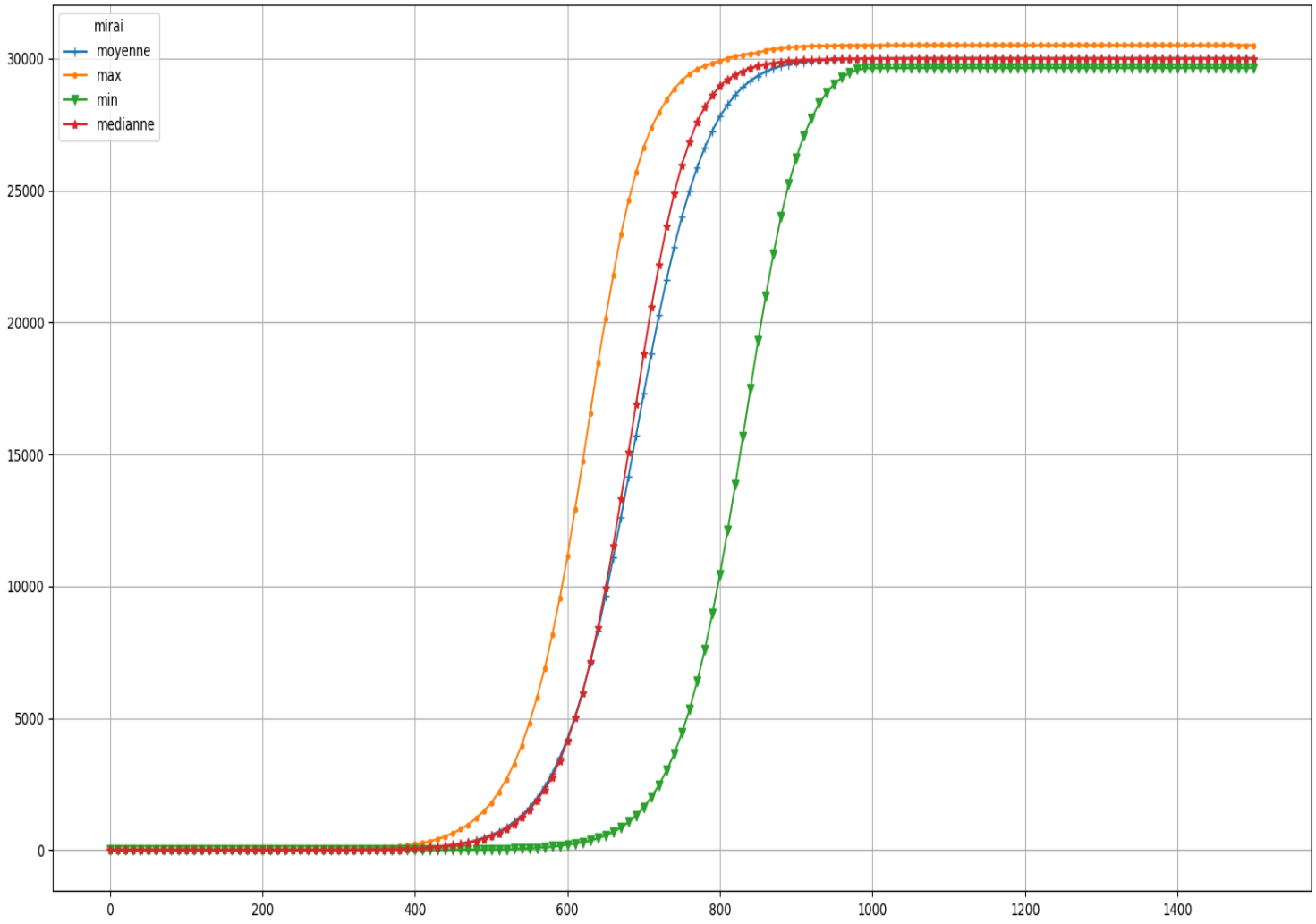

Figure 4.28 – Évolution de la population du botnet #2 sur 1500 tours

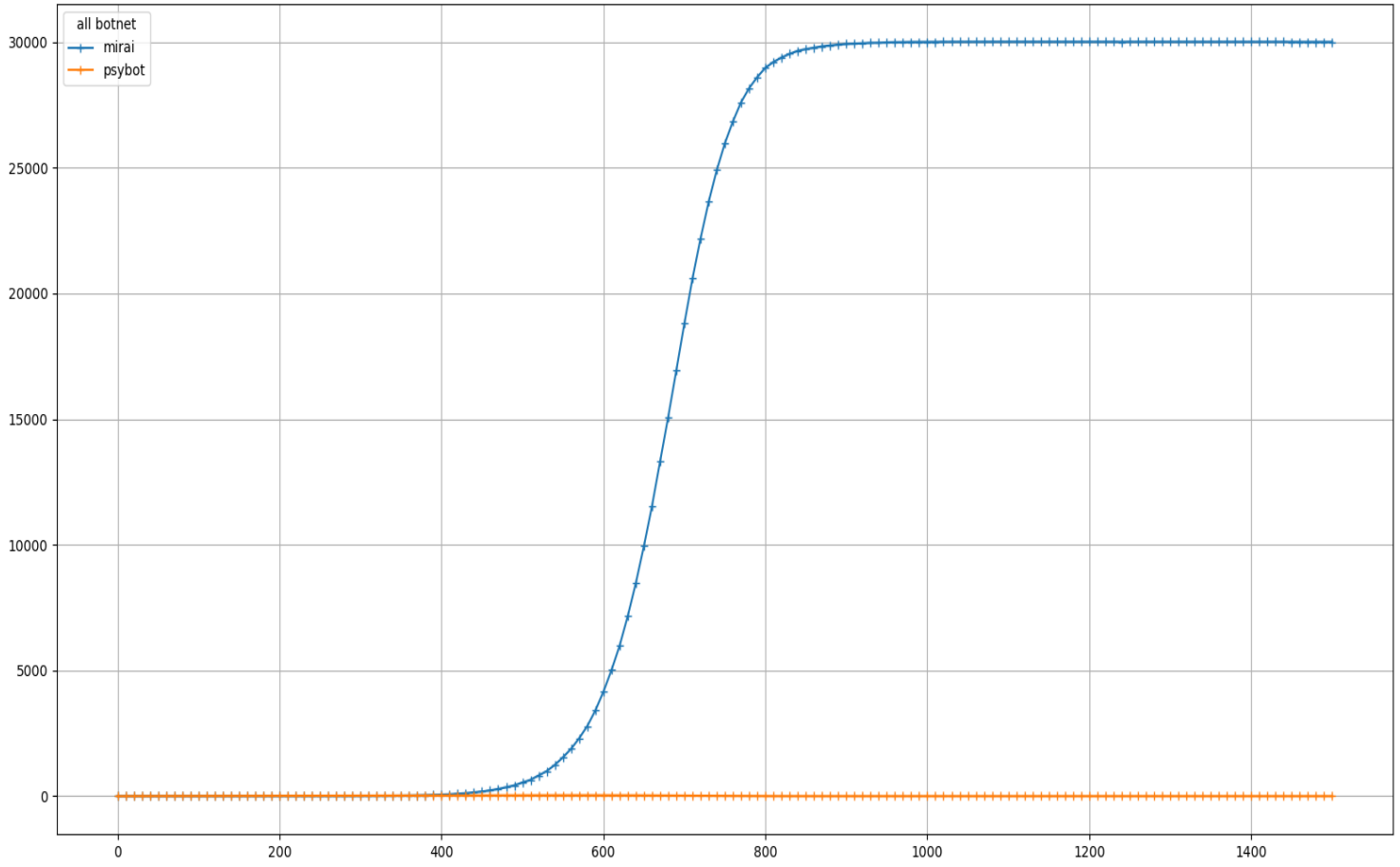

Figure 4.29 – Évolution de la population des deux botnet sur 1500 tours

L'ensemble de ces expériences à pu nous donner beaucoup de résultats pertinents, qui nous permettrons de répondre à plusieurs de nos interrogations concernant l'impact des fonctionnalités implémentées par des réseaux de zombies. Nous interpréterons et critiquerons ces résultats dans le prochain chapitre.

CHAPITRE 5

CRITIQUES ET INTERPRÉTATION

Dans ce dernier chapitre, nous allons interpréter et critiquer nos résultats. Nous exhiberons les avantages et les inconvénients de nos travaux. Enfin, à la dernière section, nous réfléchirons à de possibles nouvelles solutions afin de lutter plus efficacement contre les réseaux de zombies.

5.1 INTERPRÉTATIONS DES GRAPHES D'ÉVOLUTIONS

En analysant le graphe phylogénique, on observe une tendance à l'augmentation du nombre de fonctionnalités des réseaux de zombies. En effet, plus le rayon d'un nœud est grand, plus son nombre de fonctionnalités est élevé. Ces programmes malveillants deviennent en général de plus en plus complexes. Nous supposons que cette augmentation de la complexité provient de la réutilisation de fonctionnalités (pas nécessairement une réutilisation de code). En effet, lorsqu'un réseau fonctionne bien, des attaquants peuvent reprendre le concept et les principes de fonctionnement. Ils rajoutent alors de nouvelles fonctionnalités afin de l'améliorer. Ce type de graphe nous permet aussi de démontrer une certaine proximité des programmes. En effet, nous relevons que Chuck Norris et Psyb0t ont un nombre de fonctionnalités très proches et que la totalité des fonctionnalités de Psyb0t a été transmise à Chuck Norris. Cette proximité a été démontrée par des analyses de code Čeleda *et al.* (2010).

Nous observons aussi que des programmes plus complexes tels que Mirai ont énormément de points communs avec les logiciels plus anciens. Ce phénomène est dû à la réutilisation systématique de certaines fonctionnalités vitales pour un programme malveillant. Cela s'explique très bien avec les multi-graphes de propagation. Par exemple, à la figure 3.3, nous observons que la fonctionnalité « SYN Flood » introduite par Hydra a été réutilisée par la totalité des réseaux de zombies ayant pour objectif la création d'attaques de déni de service distribuées. Il en va de même pour la fonctionnalité « d'UDP flood » mise en place par Psyb0t. Dans cette famille de réseaux de zombies, nous pouvons observer que chaque fonctionnalité permettant la mise en place d'une attaque par déni de service a été propagée à tous les réseaux futurs.

La seule fonctionnalité n'ayant pas été propagée est l'attaque par « ICMP flood ». Cette attaque est très peu efficace et peut facilement être arrêtée par les fournisseurs d'accès CloudFlare (2019). On peut aussi observer sur la figure 3.4 le remplacement de certaines fonctionnalités comme la génération d'adresses IP par liste manuelle qui s'est transformée en liste automatique puis en génération aléatoire. Ainsi, nous observons l'apparition et l'abandon de fonctionnalités. Nous pouvons supposer qu'une fonctionnalité va se propager aux générations futures si elle est vraiment utile pour atteindre un but similaire et si la difficulté d'implémentation n'est pas trop élevée par rapport à son gain.

Par exemple, nous avons vu qu'Hydra avait transmis la fonctionnalité d'attaque « SYN Flood » à neuf des seize programmes malveillants de notre échantillon. La fonctionnalité n'a été transmise qu'à des programmes ayant pour objectifs de mettre en place des attaques de déni de service. Nous voyons qu'elle n'a pas été transmise à Darlloz qui avait pour objectif de faire miner ses victimes.

Une fonctionnalité peut disparaître si une autre plus efficace peut la remplacer. Nous pouvons observer que l'attaque par dictionnaire a progressivement été substituée par l'utilisation d'un dictionnaire pondéré et par l'exploitation de CVE. Ces deux techniques sont beaucoup plus

efficaces que la première. Nous observons le même phénomène pour les méthodes de génération d'adresses IP à scanner.

Ainsi, nous pouvons supposer que la transmission de fonctionnalité au sein des programmes malveillants va suivre une sorte de sélection naturelle basée sur l'objectif de ces programmes, l'efficacité des fonctionnalités ainsi que la difficulté d'implémentation. Les sources de certains de ces programmes étant disponibles en libre accès sur Internet, il devient très facile pour certaines fonctionnalités de se propager au travers des générations futures.

Le dernier point intéressant de cette représentation en multi-graphe de propagation est la possibilité de voir rapidement les programmes ayant introduit le plus de fonctionnalités et quels sont celles qui se sont le plus propagées.

Sur le premier graphe on peut observer que VPNFilter a introduit de nombreuses fonctionnalités d'attaques, comme par exemple la mise en place de VPN inversé, la possibilité d'espionner les réseaux locaux des appareils infectés ou encore la surveillance de système SCADA, toutes très dangereuses. Ce programme est considéré comme une menace persistante avancée (APT) mise en place par la Russie afin de nuire à l'Ukraine Cimpanu (2018b,a).

La plupart des fonctionnalités de ce ver ne se retrouveront pas dans la majorité des programmes malveillants subséquent. Il est extrêmement difficile d'implémenter ces fonctionnalités, ainsi, nous ne les verrons que dans de futures grosses attaques mises en place par d'autres gouvernements.

Sur la figure 3.4 , une simple observation montre que les fonctionnalités d'attaques contre les architectures MIPS mises en place par Hydra ont été très largement propagées. Il en est de même pour les fonctionnalités permettant de cibler les architectures ARM et x86/64 mises en place par Aidra et Carna. On peut aussi constater la proximité de certains programmes malveillants alors que leurs objectifs peuvent être opposés. Par exemple, Wifacth et Hajimé sont les deux

seuls réseaux de zombies ayant une architecture décentralisée en P2P. D'un côté, Wifatch a été créé par une équipe de « White Hat ». Lorsque Wifatch infecte un hôte, il supprime les autres programmes malveillants qu'il identifie, ferme certains ports et va afficher un message dans les logs demandant à l'utilisateur de changer son mot de passe. En conséquence, il va aider à protéger les utilisateurs. À l'opposé, Hajime va infecter ses hôtes en rajoutant des portes dérobées pouvant permettre au maître du réseau d'installer rapidement d'autres programmes malveillants. Les deux programmes utilisent des fonctionnalités proches, mais pour des objectifs opposés.

5.2 CRITIQUES ET PISTES D'AMÉLIORATION POUR LES REPRÉSENTATIONS

Notre taxonomie permet de décrire correctement les programmes étudiés et peut facilement être étendue et améliorée. Comme nous le mentionnons dans le chapitre précédent, certaines fonctionnalités existent et ont été observées dans certains réseaux de zombies (IoT ou non) mais nous ne les avons pas insérés, car aucun des programmes que nous avions étudiés ne les utilisait. Cependant, la taxonomie est faite de sorte qu'il soit possible de rajouter des niveaux et des taxons sans avoir à modifier l'organisation générale de la taxonomie.

Ainsi, une piste d'amélioration pour cette étude est d'analyser plus de programmes malveillants créant des réseaux de zombies d'objets connectés. Ainsi, nous pourrions intégrer plus de fonctionnalités dans notre taxonomie et donc avoir une idée plus précise de l'évolution de ces programmes malveillants. En effet, nous n'avons étudié que 16 familles de réseaux de zombies d'objets connectés. Or en utilisant une méthode d'analyse systématique pour les résultats académiques et techniques, nous pourrions augmenter sensiblement la quantité des informations pour étudier ces programmes.

Enfin, cette étude a été menée au début de l'année 2019. Ainsi, nous pouvons l'étendre avec

l'ensemble des vers apparus en 2019. De plus, les informations sur certains vers peu influents n'étaient que peu référencées à cette époque et en effectuant les mêmes recherches sur Google, nous avons pu découvrir d'autres programmes malveillants pouvant être ajoutés à notre étude.

L'avantage de notre représentation est de pouvoir faire un graphe pour chaque famille de fonctionnalités et ainsi avoir des graphes plus lisibles. Le seul problème est que plus le nombre de programmes étudiés augmente et plus les graphes deviennent grands et donc moins lisibles. Ainsi, il pourra être utile de compléter ces représentations avec diverses statistiques comme par exemple le temps de vie de chaque fonctionnalité ou le nombre de vers qu'ils implémentent.

5.3 INTERPRÉTATIONS DES SIMULATIONS DE PROPAGATIONS D'INFECTIONS

Notre modèle de propagation des infections permet d'observer les influences de chaque fonctionnalités sur l'efficacité des réseaux de zombies. Grâce aux simulations faites avec ce modèles, nous avons pu constater plusieurs phénomènes intéressants.

Le premier est la divisions de la taille des réseaux de zombies de manière presque proportionnelles aux nombre de réseaux de zombies de même nature. Ce genre de phénomène a pu être observé après que le code source de Mirai ait été divulgué. En effet, plusieurs groupes ont récupéré le code et l'on adapté pour créer leur propre réseaux de zombies.

Depuis ce temps, on peut observer une saturation des réseaux de zombies d'objets connectés et de plus en plus cherchent à exploiter des vulnérabilités nouvelles afin de pouvoir voler des zombies aux autres réseaux. La division n'est cependant pas parfaitement proportionnelle du au caractère aléatoire de la distributions des victimes et de la méthode de scan. On observe d'ailleurs une variation importante entre les diverses simulations, expliquant aussi les différences entre modèles et observations.

Le deuxième phénomène que nous avons pu observer est la grande différence d'efficacité entre une méthode de scan séquentiel et une méthode de scan aléatoire. La première a une croissance linéaire, tandis que la seconde présente à partir d'un moment, une croissance exponentielle. Cette différence majeure s'explique par le fait qu'avec une méthode de scan séquentielle, seul le premier zombie fait avancer l'exploration de l'espace IP. En effet, tous les nouveaux zombies recommencent à scanner depuis le début. Cette méthode de scan est très basique, intuitive et simple à mettre en place. Cependant, on voit qu'elle est très mauvaise.

Le scan aléatoire possède une croissance exponentielle car l'ensemble des nouveaux zombies se mettent eux aussi à scanner de manière aléatoire. On peut ainsi approximer cette méthode de propagation par une simple expérience, où à chaque tour, on piocherait une boule de couleur dans un sac contenant des boules blanches et des boules noires. Si la boule piochée est blanche, alors on la conserve or du sac et au tour suivant on pioche autant de fois qu'on a de boules blanches. Ici, l'espérance du nombre de boules blanches au tour t équivaut à l'espérance du nombre de zombies que possède notre réseau au tour t. On peut aisément la calculer avec une suite récursive, prenant en compte le nombre de victimes ainsi que la taille totale de la population. Cette équation est la suivante : $p_{t+1} = p_t.(\frac{V-p_t}{T} + 1)$, avec V le nombre total d'objet vulnérables, T le nombre total d'appareils et p_t l'espérance de la population au temps t.

Enfin, nous avons pu observer les effets du mécanisme de prévention des réseaux de zombies. Cet effet est très utilisé depuis l'apparition de Mirai afin de garantir un monopole des ressources des appareils exploités. Nous avons pu observer que, lorsqu'un réseau de zombie A peut exploiter une faille présente dans une population d'objets (ayant déjà été infectés ou non), que les autres réseaux de zombies ne peuvent pas exploiter, alors le réseau A va surpasser tous les autres réseaux. Il va pouvoir infecter tous les objets de la population vulnérable, et les protéger contres les autres réseaux de zombie. Ces derniers vont donc progressivement perdre de la puissance jusqu'à un minimum. Ceci est corroboré par d'autres observations faites au cours de nos travaux.

En effet, nous avons observé une augmentation graduelle du niveau de complexité des programmes malveillants dirigés contre les objets connectés. Cette augmentation de la complexité s'accompagne d'une élévation de leur dangerosité et une meilleure adaptabilité de ces programmes. De plus, nous pouvons observer que certaines caractéristiques telles que la détection d'environnement virtualisé proviennent des logiciels malveillants attaquant traditionnellement les ordinateurs classiques et les téléphones intelligents. On peut donc supposer que les programmes malveillants vont continuer à évoluer et que dans quelques années nous pourrons observer des programmes ciblant plusieurs plateformes. Cela commence déjà à se voir avec VPNFilter qui était capable d'attaquer et d'exploiter d'autres objets du réseau local de sa victime.

Notre analyse montre aussi que les attaques par dictionnaire contre les protocoles SSH et Telnet sont les plus nombreuses et font partie des stratégies les plus efficaces. Ce genre d'attaque a commencé en 2008 et est toujours utilisé. Cette stratégie a notamment permis à Mirai de mettre en place un réseau de zombies suffisamment grand pour créer la plus grande attaque de déni de service jamais enregistrée.

Une méthode simple pour protéger les objets connectés de ce genre d'attaque est de fermer les ports non utilisés et d'utiliser un mot de passe aléatoire. Cette méthode est tellement efficace que les réseaux de zombies ont commencé à l'utiliser pour sécuriser leurs hôtes afin que d'autres programmes ne puissent pas les corrompre.

Cependant, nous avons aussi constaté l'augmentation du nombre d'utilisations de CVE pour exploiter des objets connectés. Ces dernières, bien que plus complexes à implémenter, permettent une expansion plus rapide du réseau de zombies et sont bien plus efficaces qu'une utilisation de dictionnaire. Ce type d'exploit ne peut pas être stoppé par l'utilisation d'un mot de passe fort.

Cette observation permet de prédire que les divers réseaux de zombies vont exploiter de plus en plus de failles et délivrer de plus en plus de patchs de ces dernières, afin de garder leur monopole.

De plus, de manière générale, chaque exploit rajouté permet d'augmenter le nombre de victimes potentielles. Enfin, on peut prédire que dans quelques années, on observera une fusion entre réseaux de zombies traditionnel (ciblant PC et serveurs) et d'objets connectés.

En effet, lorsque la course aux exploits commencera, les réseaux devront être très modulaires afin de pouvoir constamment rajouter des modules d'exploitation de failles et les patch correspondants. Or pour créer des réseaux de zombies classiques, il existe deux grandes méthodes : diffuser des pourriels ou exploiter des failles non patchées. Ainsi, il serait idiot de ne pas rajouter dans la liste des vulnérabilités exploitées quelques exploits pour rajouter des PC et serveurs. Enfin, la tailles des réseaux de zombies d'objets connectés permet de diffuser plus largement et facilement des pourriels, pouvant ainsi aider à déployer des réseaux de zombies plus classiques.

5.4 CRITIQUES

Comme nous l'avons expliqué, notre modèle permet de rendre compte de manière précise les effets de chaque fonctionnalités jouant un rôle dans la phase de recrutement d'un réseau de zombie. Cependant, il reste encore beaucoup de comportements à implémenter. Par exemple, nous implémenterons la stratégie d'exploit avec un dictionnaire pondéré, permettant de passer moins de temps à essayer d'exploiter un appareil. Enfin, lorsque nous aurons implémenté et tester la majeur partie des fonctionnalités existantes, nous implémenterons de nouvelles stratégies comme par exemple la version distribuée de ZMAP Adrian *et al.* (2014). Le but sera de trouver des stratégies toujours plus efficaces. Il nous faudra ensuite tous les tester et déterminer la combinaison de fonctionnalité et de stratégies permettant de dominer les autres.

De plus, il nous faut maintenant réussir à déterminer le temps que prennent chacune de ces fonctionnalités pour des réseaux comme Mirai et confronter notre modèle avec la réalité. En effet, pour l'instant nous n'avons fait que des expériences théoriques, permettant de mieux comprendre

l'impact de certaines stratégies. Maintenant, il nous faut être capable de correctement paramétrer ces dernières afin de pouvoir aider à la prédiction de la propagation de réseaux de zombies.

Enfin, le dernier point négatif de notre modèle est son temps d'exécution. En effet, le modèle de jeu en tour par tour est très gourmand en temps processeurs et nécessite donc beaucoup de temps si nous souhaitons modéliser de grandes populations sur des temps longs.

5.5 DE NOUVELLES SOLUTIONS ?

Le but ultime de nos travaux est d'aider à la création de nouvelles solutions afin de diminuer les impacts néfastes des réseaux de zombies. Ainsi, nous allons détailler ici quelques idées et pistes de réflexions pouvant servir de base pour de futurs travaux.

5.5.1 PRÉVOIR LES FUTURS AVANCÉES

Afin de pouvoir combattre efficacement les impacts néfastes des réseaux de zombies, nous pensons qu'il est important de bien décrire ces réseaux et leurs évolutions. Le but est ensuite de pouvoir prévoir certaines évolutions et commencer à préparer des contre-mesures avant que ces dernières n'apparaissent. Dans la section précédente, nous avons prédit que, dans le futur, les réseaux de zombies exploiteraient de plus en plus les failles CVE pour toucher de plus en plus d'objets différents et pour augmenter leur vitesse d'exploitation. Mais, il est complexe de prévoir l'apparition de tels failles et de trouver les patchs correspondant sans trouver les failles.

A terme, il pourrait être intéressant d'observer le nombre de vulnérabilités qui seront corrigé par les réseaux de zombies et celles corrigés par les utilisateurs et les constructeurs. En effet, cela fait plus de dix ans que les mêmes failles sont toujours exploitées. On observe aussi que se sont les réseaux de zombies qui ont commencé à les corriger.

Ainsi, notre modèle n'aidera que peu à traiter cette partie. Tout au plus il pourra aider à comprendre l'utilisation de certaines failles plus que d'autres. Cela ne sera possible que pour les failles dont on connais les proportions d'objets vulnérables. Cela ne fonctionnera pas pour prédire l'apparition de vulnérabilité 0-day ou leur nature. Notre modèle encourage les chercheurs à les trouver avant les entités malveillante, mais n'aide pas à les découvrir.

Cependant, notre modèle peut aider à prévoir les évolutions d'autres techniques en rapport avec le recrutement. En effet, grâce à notre modèle, nous pouvons imaginer diverses stratégies d'exploration de l'espace IP et tester leurs efficacités. Nous pouvons aussi imaginer diverses méthodes d'infections ou d'organisation des réseaux. Par exemple, on peut imaginer plusieurs version d'un scan utilisant ZMAP Durumeric *et al.* (2014). On pourrait avoir une stratégie où chaque zombie est indépendant, une stratégie ou divise l'espace IP en un certain nombre de cycles et où l'on attribue un cycle à chaque zombie ou encore une stratégie ou chaque zombie scan non pas une adresse IP aléatoire mais un groupe d'adresse. Dans les deux derniers cas, si les stratégies s'avèrent pertinentes, notre modèle pourrait aussi aider à déterminer leurs paramètres optimaux.

5.5.2 LES RÉSEAUX DE BIENVEILLANCE

Comme nous l'avons dit plutôt, le but ultime de nos recherches est de pouvoir aider à combattre les impacts néfastes des réseaux de zombies. Cependant, plutôt que de mettre en place des méthodes curatives avec des systèmes de détection d'intrusion, nous proposons d'essayer de mettre en place une méthode préventive.

En effet, nous pensons que déterminer les meilleurs stratégies de scan, d'infection et de réplication peuvent servir à mettre en place un système de mise à jour distribué et obligatoire. Plutôt que de simplement étudier et détecter l'ensemble des stratégies des réseaux de zombies, nous

proposons d'utiliser ces fonctionnalité pour combattre ces réseaux malveillants. Ce système aurait une architecture hybride, avec des serveurs centraux contrôlés par des organismes d'état. L'ensemble des objets mis à jour formeront un réseau pair à pair pour faciliter la propagation des mise à jour et la robustesse du système.

Comme nous l'avons mentionné plus tôt, Mirai est toujours le programme malveillant le plus utilisé pour mettre en place des réseaux de zombie. Or ce dernier est actif depuis 2016 et les correctifs pour s'en prémunir sont extrêmement simple. Ceci nous montre que les utilisateurs ne mettent pas à jour leurs objets connectés et que la corruption de leurs objets à un impact suffisamment faible pour qu'ils ne cherchent pas à changer d'objets ou à les mettre à jour.

Ainsi, au lieu que la puissance de calcul volée par les réseaux de zombies ne servent à alimenter des attaques de dénis de services, nous proposons de l'utiliser afin de combattre les programmes à l'origine de ces attaques. Notre système fonctionnera comme un réseau de zombie, mais ne distribuera que des patchs et n'aura aucune charge malveillante. Ce système devra chercher l'ensemble des objets vulnérables à une faille, puis les corriger. De plus, il pourra récolter certaines informations sur les objets qu'il mettra à jour, comme l'architecture, la marque, le modèle etc. Cela permettra de pouvoir rapidement le retrouver et le mettre à jour, si une nouvelle faille l'affectant est découverte subséquemment.

Ce système de mise à jour comprendrait deux grandes phases. La première correspond au déploiement initial. Ici,le système utilisera les meilleurs techniques de scan, d'infection et de duplication afin de pouvoir s'installer dans un maximum d'objets connectés. En s'installant, il corrigera l'ensemble des vulnérabilités connues pour un appareil et devra faire une mise à jour du firmware de l'objet afin d'être très difficilement désinstallable. Lors de cette phase, chaque objet nouvellement rajouté au système de mise à jour, se mettra lui aussi à chercher des objets vulnérables afin que la couverture de notre système soit maximale.

Lorsque l'ensemble des adresses IP auront été scannées plusieurs fois sans que de nouveaux objets ne rentre dans notre système, ce dernier se mettra en veille. Bien sûr, chaque objet aura un identifiant unique et ne recevra et n'installera que des mise à jour signée par notre système, à l'aide d'un mécanisme de signature électronique robuste.

Aussi, chaque objet transmettra des informations le concernant au système central. Cela permettra d'organiser la seconde phase de vie de notre système. Cette dernière phase correspond à la vie de notre système et sera répétée à chaque fois qu'une faille sera détecter et qu'il faudra propager son correctif. Ici, il ne sera plus question d'utiliser une méthode de scan pseudo aléatoire, mais une propagation organisée et informée. En effet, grâce aux données récoltées, nous pourrons aisément établir un ordre de propagation des correctifs, basées sur l'importance de chaque objets. Ce classement peut être similaire un algorithme de « Page-Rank » comme celui mis en place par les divers moteurs de recherches.

Une fois l'ordre de mise à jour généré, les serveurs centraux distribueront rapidement le correctif aux objets les plus important, ainsi qu'une liste des autres objets à patcher. Ainsi, la propagation serait extrêmement rapide et la puissance que pourrait obtenir un réseau de zombie utilisant cette faille, diminuerait de manière exponentielle. Enfin, si un serveur avec une connexion 10 Gbps peut scanner l'ensemble des adresses IP en moins de cinq minutes Adrian *et al.* (2014), alors une dizaine d'entre eux, ainsi qu'un réseaux pair à pair devrait être capable de propager un correctif en quelques minutes. Une telle vitesse de propagation permettrait de rendre impossible l'utilisation d'une faille associée à une patch pour créer un réseau de zombie.

Actuellement, il existe quelques projet ayant un objectif similaire, comme par exemple AntibIoTic De Donno *et al.* (2018a) ou le projet de Jerkins et al. Jerkins (2017). Le problème actuel de ce genre de projet, est qu'ils sont complètement illégaux dans la majeure partie des pays. De plus, ces derniers vont en général récupérer le code d'un programme malveillant et supprimer la partie malveillante pour ensuite aider à combattre ce même vers. Cela n'est pas forcément

la méthode la plus efficace et nous proposons l'idée d'un système basé sur les évolutions des programmes malveillants afin de créer le système le plus efficace possible. Cependant, ce projet n'est aujourd'hui qu'une idée et son efficacité supposée n'est encore que purement spéculative. Il nous faut encore perfectionner notre modèle pour ensuite pouvoir simuler la mise en place d'un tel système de mise à jour.

CONCLUSION

Dans ce mémoire, nous avons analyser la sécurité fournies par divers protocole de communication pour objets connectés. Nous avons remarqué, que sous certaines conditions particulière, ces dernières pouvaient fournir une protection suffisante. Nous avons aussi remarqué que l'ensemble des dommages provoqués par le manque de sécurités des objets connectés ne venaient pas de ces protocoles, mais plutôt de l'exploitation d'objets directement connectés au réseau Internet.

Ainsi, nous avons identifié comme problème majeur la formation et l'utilisation de réseaux de zombies d'objets connectes. Afin de trouver de nouvelles pistes pour combattre ces réseaux malveillants, nous avons développé deux outils d'analyse et de modélisation afin de mieux comprendre l'évolution de ces programmes.

Notre première contribution est une taxonomie qui permet de mieux décrire les réseaux de zombies composés d'objets connectés. Celle-ci, basée sur les fonctionnalités utilisées par les réseaux de zombies observés, permet de créer une nouvelle représentation des évolutions des programmes malveillants. En effet, la représentation sous forme de graphe de propagation permet de rapidement détecter les fonctionnalités intégrées dans divers réseaux de zombies. Cela permet aussi de rapidement identifier les fonctionnalités les moins efficaces, très peu propagées aux travers des réseaux étudiés.

Afin de mieux comprendre pourquoi certaines fonctionnalités ont tendance à disparaître et d'autres à rester, nous avons développé un modèle permettant de simuler de manière fine l'impact de diverses fonctionnalités sur l'efficacité des réseaux de zombies. Ce modèle simule, sous forme d'un jeu en tour par tour, le phénomène de propagation des réseaux de zombies.

Cela nous a permis de modéliser plusieurs stratégies et de montrer leurs effets sur la vitesse de propagation des infections. En effet, nous avons pu modéliser et comparer les stratégies de scan séquentiel et aléatoire. Nous avons aussi réussi à mettre en compétition ces deux stratégies. Nos résultats montre que le scan aléatoire est bien plus efficace que le scan séquentiel. Ceci nous permet de partiellement confirmer notre hypothèse voulant que les fonctionnalités les plus efficaces ont tendance à mieux ce propager.

Nos modèles sont encore améliorable et il reste encore de nombreuses expériences à faire afin de confirmer totalement nos hypothèses concernant l'évolution des réseaux de zombies. Cependant, nos modèles peuvent déjà être utilisé afin de simuler de nouveaux comportement, n'ayant pas été observé. Cela peut nous permettre d'anticiper les évolutions future des réseaux de zombies. De plus cela ouvre la voie vers de nouveaux système de corrections de failles. En effet, nous avons évoqué l'idée d'un système de mise à jour distribué et obligatoire afin de combattre les réseaux de zombie. Nos futur travaux concerneront l'amélioration de nos modèles ainsi que la modélisation d'un tel système de mise à jour. Enfin, nous évoquerons les aspects légaux et éthiques de ce projet.

Pour conclure, nous répétons, comme de nombreux chercheurs avant nous, que les configurations et les mots de passe par défauts, connus de tous, sont le problème majeur de la sécurité des objets connectés. Le second est le faible taux de mise à jour des appareils, qui ne sont pas maintenus par les constructeurs, ou difficile à mettre à jour par les utilisateurs. Cela entraîne l'apparition de réseaux de zombies, exploitant des failles découverte il y a plus de dix ans. Il est donc important de se concentrer sur ces problèmes, et d'adopter des stratégies cohérentes pour

lutter contre les effets indésirables de ce manque de sécurité. C'est dans cette optique que nous avons évoqué la piste d'un système de mise à jour obligatoire, géré par des agences d'état et contrôlé par les sociétés civiles.

BIBLIOGRAPHIE

Abaid, Z., Sarkar, D., Kaafar, M. A. et Jha, S. (2016). The early bird gets the botnet : a markov chain based early warning system for botnet attacks. Dans *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, 61–68. IEEE.

Adrian, D., Durumeric, Z., Singh, G. et Halderman, J. A. (2014). Zippier zmap : Internet-wide scanning at 10 gbps. Dans *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, San Diego, CA. USENIX Association.

Akamai (2017). Whitepaper : Dns reflection, amplification, dns water-torture. `https://www.akamai.com/it/it/multimedia/documents/technical-publication/dns-reflection-vs-dns-mirai-technical-publication.pdf`.

Alemdar, H. et Ersoy, C. (2010). Wireless sensor networks for healthcare : A survey. *Computer Networks*, *54*(15), 2688 – 2710. `http://dx.doi.org/https://doi.org/10.1016/j.comnet.2010.05.003`. Récupéré de `http://www.sciencedirect.com/science/article/pii/S1389128610001398`

Aliance, Z. (2017). Zigbee : Securing the wireless iot.

Alliance, Z. (2014). Zigbee specifications. http://www.zigbee.org/wp-content/uploads/2014/11/docs-05-3474-20-0csg-zigbee-specification.pdf.

Angrishi, K. (2017). Turning internet of things(iot) into internet of vulnerabilities (iov) : Iot botnets. *arXiv preprint arXiv :1702.03681*.

ANSSI (2014). Référentiel général de sécurité, version 2.0. https://www.ssi.gouv.fr/uploads/2014/11/RGS_v-2-0_B1.pdf.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K. et Zhou, Y. (2017). Understanding the mirai botnet. Dans *26th USENIX Security Symposium (USENIX Security 17)*, 1093–1110., Vancouver, BC. USENIX Association. Récupéré le 2017 de https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

Anubhav, A. (2017). Huawei router exploit involved in satori and brickerbot given away for free on christmas by blackhat santa. https://blog.newskysecurity.com/huawei-router-exploit-involved-in-satori-and-brickerbot-given-away-for-free-on-christ

Badenhop, C. W., Graham, S. R., Ramsey, B. W., Mullins, B. E. et Mailloux, L. O. (2017). The z-wave routing protocol and its security implications. *Comput. Secur.*, *68*(C), 112–129. http://dx.doi.org/10.1016/j.cose.2017.04.004. Récupéré de https://doi.org/10.1016/j.cose.2017.04.004

Ballano, M. (2015). Is there an Internet-of-things vigilante out there ? https://www.symantec.com/connect/blogs/there-internet-things-vigilante-out-there?SID=skim66960X1514734X27e68170b54f265132adac9b5f1f4e33&API1=100&API2=7596969&cjid=7596969.

Benjamin, V., Raphael, K. et Sylvain, H. (2019). 10 years of iot malware : A feature-based taxonomy. Dans *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 458–465. http://dx.doi.org/10.1109/QRS-C.2019.00088

Bharathi, H., Srivani, U., Azharudhin, M. D., Srikanth, M. et Sukumarline, M. (2017). Home automation by using raspberry pi and android application. Dans *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, volume 2, 687–689. http://dx.doi.org/10.1109/ICECA.2017.8212754

Botticelli, B. (2017). IoT honeypots : State of the art. https://fr.slideshare.net/BiagioBotticelli/state-of-the-art-iot-honeypots.

Buterin, V. (2019). What is ethereum ? https://ethereum.org/beginners/.

Carna (2012). Internet census 2012, port scanning /0 using insecure embedded devices. http://census2012.sourceforge.net/paper.html.

Celebucki, D., Lin, M. A. et Graham, S. (2018). A security evaluation of popular internet of things protocols for manufacturers. Dans *2018 IEEE International Conference on Consumer Electronics (ICCE)*, 1–6. http://dx.doi.org/10.1109/ICCE.2018.8326099

Celeda, P., Krejci, R., Vykopal, J. et Drasar, M. (2010). Embedded malware - an analysis of the Chuck Norris botnet. Dans *2010 European Conference on Computer Network Defense*, 3–10. http://dx.doi.org/10.1109/EC2ND.2010.15

Chase, J. (2013). The evolution of the internet of things. http://www.ti.com/lit/ml/swrb028/swrb028.pdf.

Chen, Z., Gao, L. et Kwiat, K. (2003). Modeling the spread of active worms. Dans *IEEE*

INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428), volume 3, 1890–1900. IEEE.

Chen, Z. et Ji, C. (2005). Spatial-temporal modeling of malware propagation in networks. *IEEE Transactions on Neural networks*, *16*(5), 1291–1303.

Cimpanu, C. (2017). BrickerBot dev claims cyber-attack that affected over 60,000 Indian modems. https://www.bleepingcomputer.com/news/security/brickerbot-dev-claims-cyber-attack-that-affected-over-60-000-indian-modems/.

Cimpanu, C. (2018a). FBI takes control of APT28's VPNFilter botnet. https://www.bleepingcomputer.com/news/security/fbi-takes-control-of-apt28s-vpnfilter-botnet/.

Cimpanu, C. (2018b). Ukraine says it stopped a VPNFilter attack on a chlorine distillation station. https://www.bleepingcomputer.com/news/security/ukraine-says-it-stopped-a-vpnfilter-attack-on-a-chlorine-distillation-station/.

Claud, X. et Cong, Z. (2017). New IoT/Linux malware targets dvrs, forms botnet. https://unit42.paloaltonetworks.com/unit42-new-iotlinux-malware-targets-dvrs-forms-botnet/.

CloudFlare (2019). Ping (icmp) flood DDoS attack. https://www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack/.

CloudFlare (2020). What is ad fraud ? | ad click fraud. https://www.cloudflare.com/learning/bots/what-is-ad-fraud/.

cyberveille sante.gouv.fr (2018). Z-shave, une attaque qui pourrait affecter plus de 100 millions d'appareils utilisant le protocole de communica-

tion z-wave. https://www.cyberveille-sante.gouv.fr/cyberveille/782-z-shave-une-attaque-qui-pourrait-affecter-plus-de-100-millions-dappareils-util

Dagon, D., Gu, G., Lee, C. P. et Lee, W. (2007). A taxonomy of botnet structures. Dans *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, 325–339. http://dx.doi.org/10.1109/ACSAC.2007.44

Daley, D. et Gani, J. (1999). Epidemic modeling : An introduction cambridge university press. *NY, New York*, *228*.

Daniel, S. (2018). Iot botnets on the rise. https://blog.radware.com/security/2018/10/iot-botnets-on-the-rise/.

De Donno, M., Dragoni, N., Giaretta, A. et Mazzara, M. (2018a). Antibiotic : Protecting iot devices against ddos attacks. Dans P. Ciancarini, S. Litvinov, A. Messina, A. Sillitti, et G. Succi (dir.). *Proceedings of 5th International Conference in Software Engineering for Defence Applications*, 59–72., Cham. Springer International Publishing.

De Donno, M., Dragoni, N., Giaretta, A. et Spognardi, A. (2018b). DDoS-capable IoT malwares : Comparative analysis and Mirai investigation. *Security and Communication Networks*, 1–30. http://dx.doi.org/10.1155/2018/7178164

DroneBL (2009). DrobeBL. https://dronebl.org/blog/8.

Drozhzhin, A. (2015). Anti-spam : Comment protéger votre boîte de réception ? https://www.kaspersky.fr/blog/kaspersky-anti-spam-protection/4600/.

Durumeric, Z., Adrian, D., Mirian, A., Bailey, M. et Halderman, J. A. (2015). A search engine backed by internet-wide scanning. Dans *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, 542–553., New York,

NY, USA. ACM. http://dx.doi.org/10.1145/2810103.2813703. Récupéré de http://doi.acm.org/10.1145/2810103.2813703

Durumeric, Z., Bailey, M. et Halderman, J. A. (2014). An internet-wide view of internet-wide scanning. Dans *23rd USENIX Security Symposium (USENIX Security 14)*, 65–78., San Diego, CA. USENIX Association.

Dworkin, M. (2004). *Recommendation for block cipher modes of operation : The CCM mode for authentication and confidentiality*. Rapport technique, National Institute of Standards and Technology.

Dyn DNS (2016). Incident report for oracle + dyn. https://www.dynstatus.com/incidents/nlr4yrr162t8.

Edmund, B. (2018). VPNFilter iii : More tools for the Swiss army knife of malware. https://blog.talosintelligence.com/2018/09/vpnfilter-part-3.html.

Ehrlich, D. (2008). Sigma designs buying smart network chipmaker zensys. https://gigaom.com/2008/12/18/sigma-designs-buying-smart-network-chipmaker-zensys/.

Evans, D. (2011). L'internet des objets comment l'évolution actuelle d'internet transforme-t-elle le monde ? Récupéré de https://www.cisco.com/c/dam/global/en_ca/solutions/executive/assets/pdf/internet-of-things-fr.pdf

Farahani, B., Firouzi, F., Chang, V., Badaroglu, M., Constant, N. et Mankodiya, K. (2018). Towards fog-driven iot ehealth : Promises and challenges of iot in medicine and healthcare. *Future Generation Computer Systems*, *78*, 659 – 676. http://dx.doi.org/https://doi.org/10.1016/j.future.2017.04.036

Fisher, D. (2013). Qu'est-ce q'un botnet ? https://www.kaspersky.fr/blog/quest-ce-quun-botnet/888/.

FortiGuard (2017). Reaper : The next evolution of IoT botnets. https://www.fortinet.com/blog/threat-research/reaper-the-next-evolution-of-iot-botnets.html.

Geenens, P. (2017). Brickerbot – the dark knight of iot. https://blog.radware.com/security/2017/04/brickerbot-dark-knight-iot/.

Guarnizo, J. D., Tambe, A., Bhunia, S. S., Ochoa, M., Tippenhauer, N. O., Shabtai, A. et Elovici, Y. (2017). Siphon : Towards scalable high-interaction physical honeypots. Dans *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, CPSS '17, 57–68., New York, NY, USA. ACM. http://dx.doi.org/10.1145/3055186.3055192. Récupéré de http://doi.acm.org/10.1145/3055186.3055192

Hachem, N., Ben Mustapha, Y., Granadillo, G. G. et Debar, H. (2011). Botnets : Lifecycle and taxonomy. Dans *2011 Conference on Network and Information Systems Security*, 1–8. http://dx.doi.org/10.1109/SAR-SSI.2011.5931395

Hayashi, K. (2013). Linux.darlloz. https://www.symantec.com/security-center/writeup/2013-112710-1612-99.

Hayashi, K. (2014). IoT worm used to mine cryptocurrency. https://www.symantec.com/connect/blogs/iot-worm-used-mine-cryptocurrency.

Hu, S., Wang, H., She, C. et Wang, J. (2011). Agont : Ontology for agriculture internet of things. Dans D. Li, Y. Liu, et Y. Chen (dir.). *Computer and Computing Technologies in Agriculture IV*, 131–137., Berlin, Heidelberg. Springer Berlin Heidelberg.

ifding (2017). GitHub repo of open source IoT malware. https://github.com/ifding/iot-malware.

ISO (1978). Modèle osi. https://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip.

ISO (2019). Iso/iec 18033-6 :2019(en) it security techniques — encryption algorithms — part 6 : Homomorphic encryption. https://www.iso.org/obp/ui/#iso:std:iso-iec:18033:-6:ed-1:v1:en.

Janus, M. (2011). Heads of the Hydra. malware for network devices. https://securelist.com/heads-of-the-hydra-malware-for-network-devices/36396/.

Jerkins, J. A. (2017). Motivating a market or regulatory solution to iot insecurity with the mirai botnet code. Dans *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, 1–5. IEEE.

Ji, Y., Yao, L., Liu, S., Yao, H., Ye, Q. et Wang, R. (2018). The study on the botnet and its prevention policies in the internet of things. Dans *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, 837–842. http://dx.doi.org/10.1109/CSCWD.2018.8465280

Ji, Y., Yao, L., Liu, S., Yao, H., Ye, Q. et Wang, R. (2018). The study on the botnet and its prevention policies in the internet of things. Dans *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, 837–842. IEEE.

Kambourakis, G., Kolias, C. et Stavrou, A. (2017). The mirai botnet and the iot zombie armies. Dans *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, 267–272. http://dx.doi.org/10.1109/MILCOM.2017.8170867

Kerckhoffs, A. (1883). La cryptographie millitaire. *Journal des Sciences Millitaires*, *IX*.

Kevin D.Mitnick, W. L. S. (2002). *The Art of Deception. Controlling the human element of security.*

Kolias, C., Kambourakis, G., Stavrou, A. et Voas, J. (2017). DDoS in the IoT : Mirai and other botnets. *Computer*, *50*(7), 80–84. http://dx.doi.org/10.1109/MC.2017.201

Koroniotis, N., Moustafa, N. et Sitnikova, E. (2019). Forensics and deep learning mechanisms for botnets in internet of things : A survey of challenges and solutions. *IEEE Access*, *7*, 61764–61785. http://dx.doi.org/10.1109/ACCESS.2019.2916717

Krebs, B. (2017). Fear the reaper, or reaper madness ? https://krebsonsecurity.com/2017/10/fear-the-reaper-or-reaper-madness/#more-41321.

Labbe, P. (2017). Connaissez-vous les chaussons connectés. https://www.objetconnecte.net/chaussons-connectes-selection/.

Labs, B. L. (2018). A new phase of themoon.

Leyden, J. (2017). 'amnesia' IoT botnet feasts on year-old unpatched vulnerability. https://www.theregister.co.uk/2017/04/07/amnesia_iot_botnet/.

Lueth, K. L. (2018). State of the iot 2018 : Number of iot devices now at 7b – market accelerating. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

Malik, M. et M.Léveillé, M.-E. (2016). Meet Remaiten – a Linux bot on steroids targeting routers and potentially other IoT devices.

Malwarebytes (2019). Malware. https://www.malwarebytes.com/malware/.

Manoharan, S. (2018). Iot malware :an analysis of device hijacking.

Marzano, A., Alexander, D., Fonseca, O., Fazzion, E., Hoepers, C., Steding-Jessen, K., Chaves, M., Cunha, I., Guedes, D. et Meira Jr, W. (2018). The evolution of Bashlite and Mirai IoT botnets. http://dx.doi.org/10.1109/ISCC.2018.8538636

Molisch, A. F., Balakrishnan, K., Chong, C.-C., Emami, S., Fort, A., Karedal, J., Kunisch, J., Schantz, H., Schuster, U. et Siwiak, K. (2004). Ieee 802.15. 4a channel model-final report. *IEEE P802*, *15*(04), 0662.

Moore, D., Shannon, C., Voelker, G., Savage, S. *et al.* (2004). *Network telescopes : Technical report*. Rapport technique, Cooperative Association for Internet Data Analysis (CAIDA).

Moore, D., Shannon, C., Voelker, G. M. et Savage, S. (2003). Internet quarantine : Requirements for containing self-propagating code. Dans *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 3, 1901–1910. IEEE.

Morgner, P., Mattejat, S., Benenson, Z., Müller, C. et Armknecht, F. (2017). Insecure to the touch : Attacking ZigBee 3.0 via touchlink commissioning. Dans *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '17, 230–240., New York, NY, USA. ACM. http://dx.doi.org/10.1145/3098243.3098254. Récupéré de http://doi.acm.org/10.1145/3098243.3098254

Nakamoto, S. (2008). Bitcoin whitepaper. *URL : https ://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)*.

Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T. et Rossow, C. (2015). Iotpot : Analysing the rise of iot compromises. Dans *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, Washington, D.C. USENIX Association. Récupéré de https://www.usenix.org/conference/woot15/workshop-program/presentation/pa

Point, C. (2017). Iotroop botnet : The full investigation. https://research.checkpoint.com/iotroop-botnet-full-investigation/.

Provos, N. *et al.* (2004). A virtual honeypot framework. Dans *USENIX Security Symposium*, volume 173, 1–14.

Radaware (2017). "BrickerBot" results in PDoS attack. https://security.radware.com/ddos-threats-attacks/brickerbot-pdos-permanent-denial-of-service/.

Rawat, R. S., Pilli, E. S. et Joshi, R. C. (2018). Survey of peer-to-peer botnets and detection frameworks. *IJ Network Security*, *20*(3), 547–557.

Ronen, E., Shamir, A., Weingarten, A. et O'Flynn, C. (2017). Iot goes nuclear : Creating a zigbee chain reaction. Dans *2017 IEEE Symposium on Security and Privacy (SP)*, 195–212. http://dx.doi.org/10.1109/SP.2017.14

Ropert, S. (2017). Iot : Un marché de 36 milliards d'objets connectés à internet d'ici 2030. https://fr.idate.org/marche-internet-des-objets/.

Rouch, L., François, J., Beck, F. et Lahmadi, A. (2017). A universal controller to take over a z-wave network.

Rozier, U. (2017). 14 nouveautés du ces 2017, le pire du pire des objets connectés.

Saint-Laurent, M. (2017). IoT : Plus de 5 millions de produits achetés et 1 milliard d'euros de revenu généré en 2017. https://www.gfk.com/fr/insights/press-release/iot-plus-de-5-millions-de-produits-achetes-et-1-milliardd'eurosde-revenu-genere-en-2017/.

Salman, T. et Jain, R. (2015). Networking protocols and standards for internet of things. *Internet of Things and Data Analytics Handbook*, *2015*, 215–238.

Sam Edwards, I. P. (2016). Hajime : Analysis of a decentralized internet worm for iot devices.

Schmeiser, A. F. (2017). Internet of things–related business change : A qualitative meta-analysis.

Schneier, B. (1995). *Applied Cryptography (2nd Ed.) : Protocols, Algorithms, and Source Code in C*. USA : John Wiley Sons, Inc.

Scott Sr, J. et Summit, W. (2016). Rise of the machines : The dyn attack was just a practice run december 2016.

Soltan, S., Mittal, P. et Poor, H. V. (2018). Blackiot : Iot botnet of high wattage devices can disrupt the power grid. Dans *27th USENIX Security Symposium (USENIX Security 18)*, 15–32., Baltimore, MD. USENIX Association. Récupéré de https://www.usenix.org/conference/usenixsecurity18/presentation/soltan

Staniford, S., Paxson, V., Weaver, N. *et al.* (2002). How to own the internet in your spare time. Dans *USENIX security symposium*, volume 2, 14–15.

Stojkoska, B. L. R. et Trivodaliev, K. V. (2017). A review of Internet of Things for smart home : Challenges and solutions. *Journal of Cleaner Production*, *140*, 1454–1464. Récupéré de http://www.sciencedirect.com/science/article/pii/S095965261631589Xhttp://dx.doi.org/https://doi.org/10.1016/j.jclepro.2016.10.006

System, T. M. (2018). New rapidly-growing iot botnet - reaper. https://success.trendmicro.com/solution/1118928-new-rapidly-growing-iot-botnet-reaper#collapseTwo.

team, T. W. (2016). Wifatch gitlab. https://gitlab.com/rav7teif/linux.wifatch/.

Tierney, A. (2018). Z-Shave. exploiting Z-Wave downgrade attacks. https://www.pentestpartners.com/security-blog/z-shave-exploiting-z-wave-downgrade-attacks/.

Tuttlebee, W. H. (2003). *Software defined radio : enabling technologies*. John Wiley & Sons.

unlnow (2015). Kaiten source code. https://github.com/shipcod3/IRC-Bot-Hunters/blob/master/malicious_samples/kaiten.c.

Wainwright, P. et Kettani, H. (2019). An analysis of botnet models. Dans *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*, 116–121. ACM.

Wang, A., Chang, W., Chen, S. et Mohaisen, A. (2018). Delving into internet ddos attacks by botnets : Characterization and analysis. *IEEE/ACM Transactions on Networking (TON)*, *26*(6), 2843–2855.

Wang, A., Liang, R., Liu, X., Zhang, Y., Chen, K. et Li, J. (2017). An inside look at iot malware. Dans F. Chen et Y. Luo (dir.). *Industrial IoT Technologies and Applications*, 176–186., Cham. Springer International Publishing.

William, L. (2018a). New VPNFilter malware targets at least 500k networking devices worldwide. https://blog.talosintelligence.com/2018/05/VPNFilter.html.

William, L. (2018b). VPNFilter update - VPNFilter exploits endpoints, targets new devices. https://blog.talosintelligence.com/2018/06/vpnfilter-update.html.

Williams, R. (2014). https ://www.telegraph.co.uk/technology/internet-security/10579677/more-than-750000-spam-emails-sent-from-fridges-and-tvs.html. https://www.telegraph.co.uk/technology/internet-security/10579677/More-than-750000-spam-emails-sent-from-fridges-and-TVs.html.

Xueqi, F., Fransisca, S., William, L. et Shangyan, L. (2017). Security analysis of ZigBee. Récupéré de https://pdfs.semanticscholar.org/3d1d/5a51d05cde08b6e52afd5bd7bc325b487a10.pdf

Yassein, M. B., Mardini, W. et Almasri, T. (2018). Evaluation of security regarding z-wave wireless protocol. Dans *Proceedings of the Fourth International Conference on Engineering & MIS 2018*, ICEMIS '18, 32 :1–32 :8., New York, NY, USA. ACM. http://dx.doi.org/10.1145/3234698.3234730. Récupéré de http://doi.acm.org/10.1145/3234698.3234730

yegenshen (2017). IoT_reaper : A rappid spreading new iot botnet.

Zaddach, Costin, J. (2018). IoT malware comprehensive survey, analysis framework and case studies. Dans *Black Hat 2018*.

Zanella, A., Bui, N., Castellani, A., Vangelista, L. et Zorzi, M. (2014). Internet of things for smart cities. *IEEE Internet of Things Journal*, *1*(1), 22–32. http://dx.doi.org/10.1109/JIOT.2014.2306328

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

Čeleda, P., Krejčí, R., Vykopal, J. et Drašar, M. (2010). Embedded malware - an analysis of the chuck norris botnet. Dans *2010 European Conference on Computer Network Defense*, 3–10. http://dx.doi.org/10.1109/EC2ND.2010.15

Ďurfina, L., Křoustek, J. et Zemek, P. (2013). Psybot malware : A step-by-step decompilation case study. Dans *2013 20th Working Conference on Reverse Engineering (WCRE)*, 449–456. http://dx.doi.org/10.1109/WCRE.2013.6671321