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.

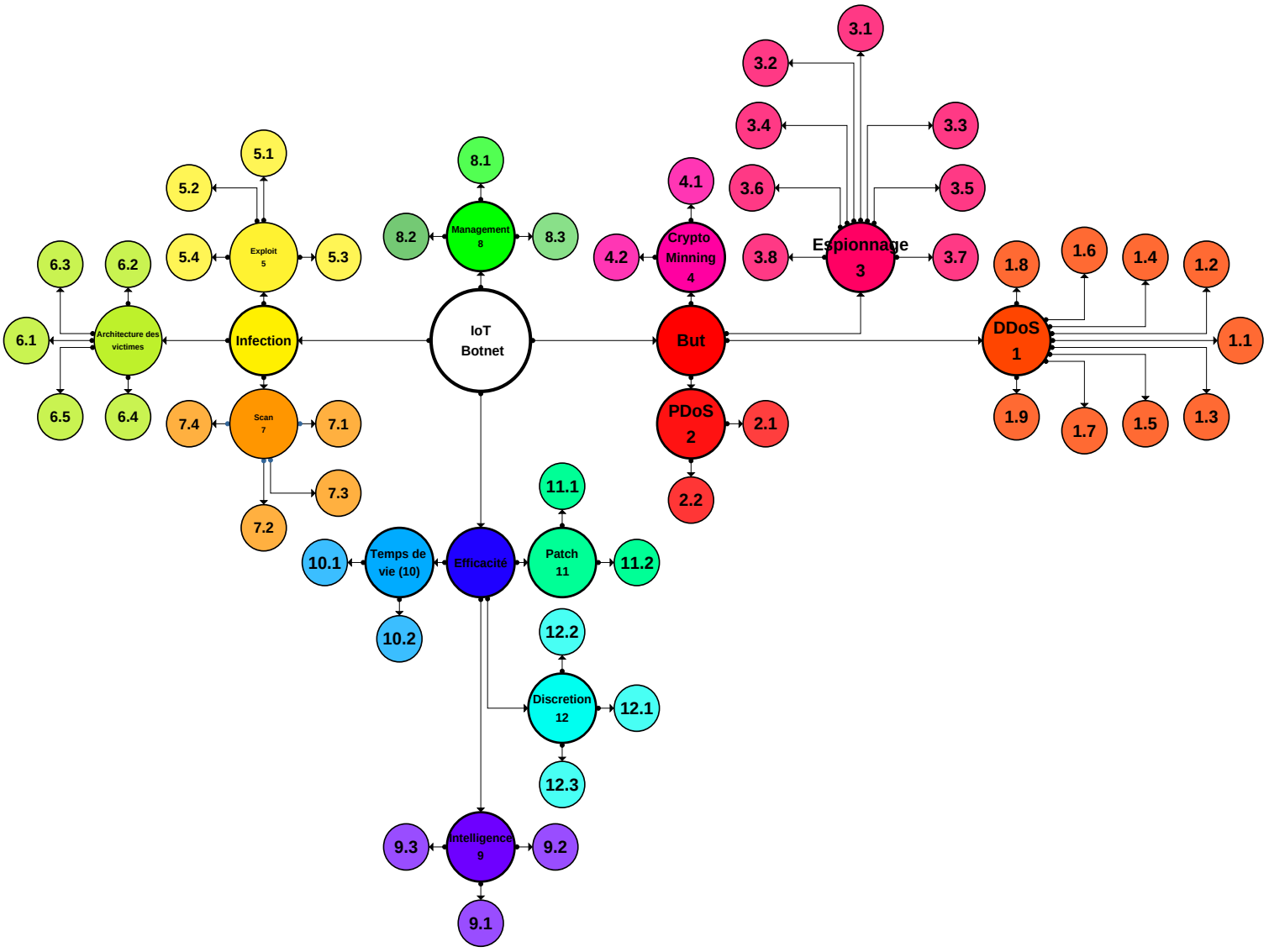

Figure 3.1 – Taxonomie des réseaux de zombies IoT

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.

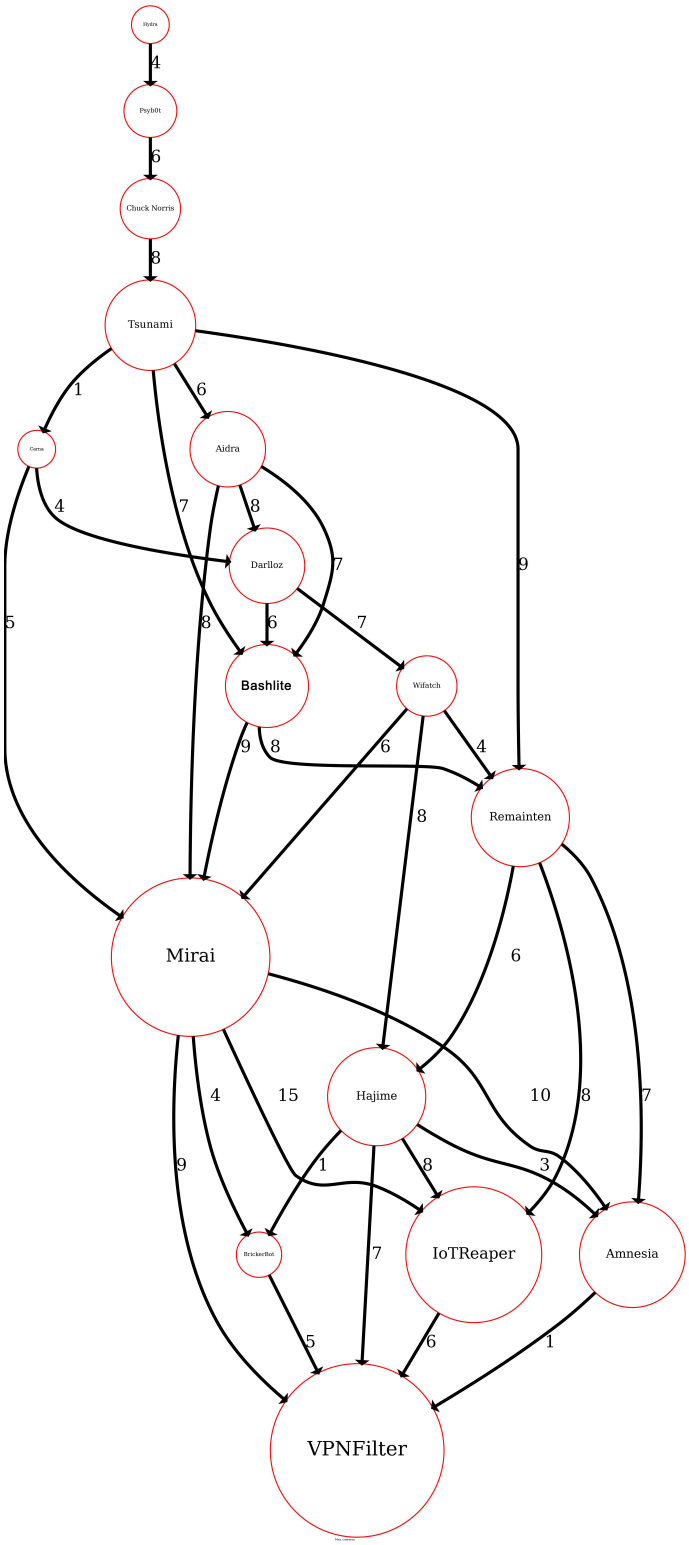

Figure 3.2 – Graphe Phylogénique des programmes malveillants.

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.

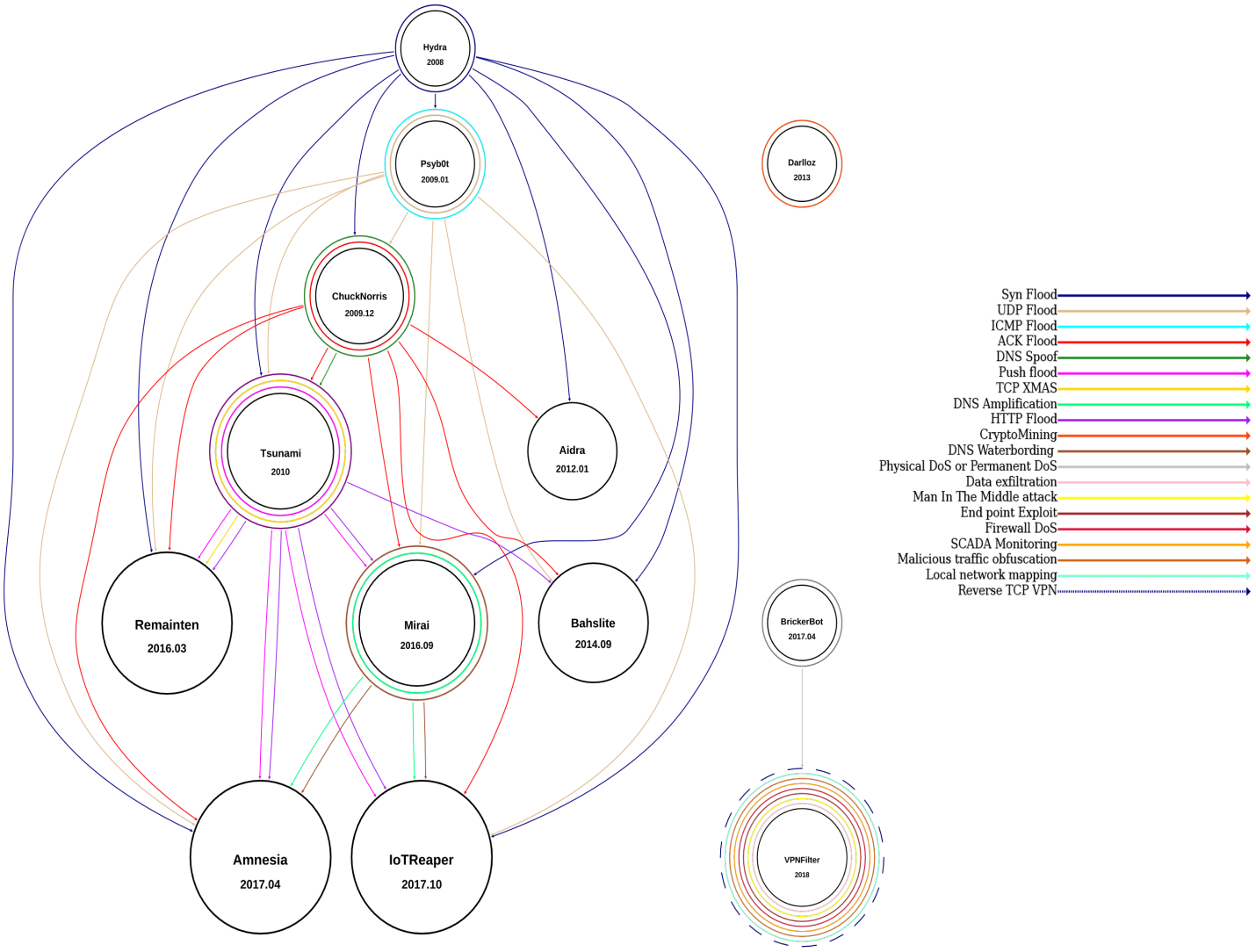

Figure 3.3 – Propagation des fonctionnalités d'attaque.

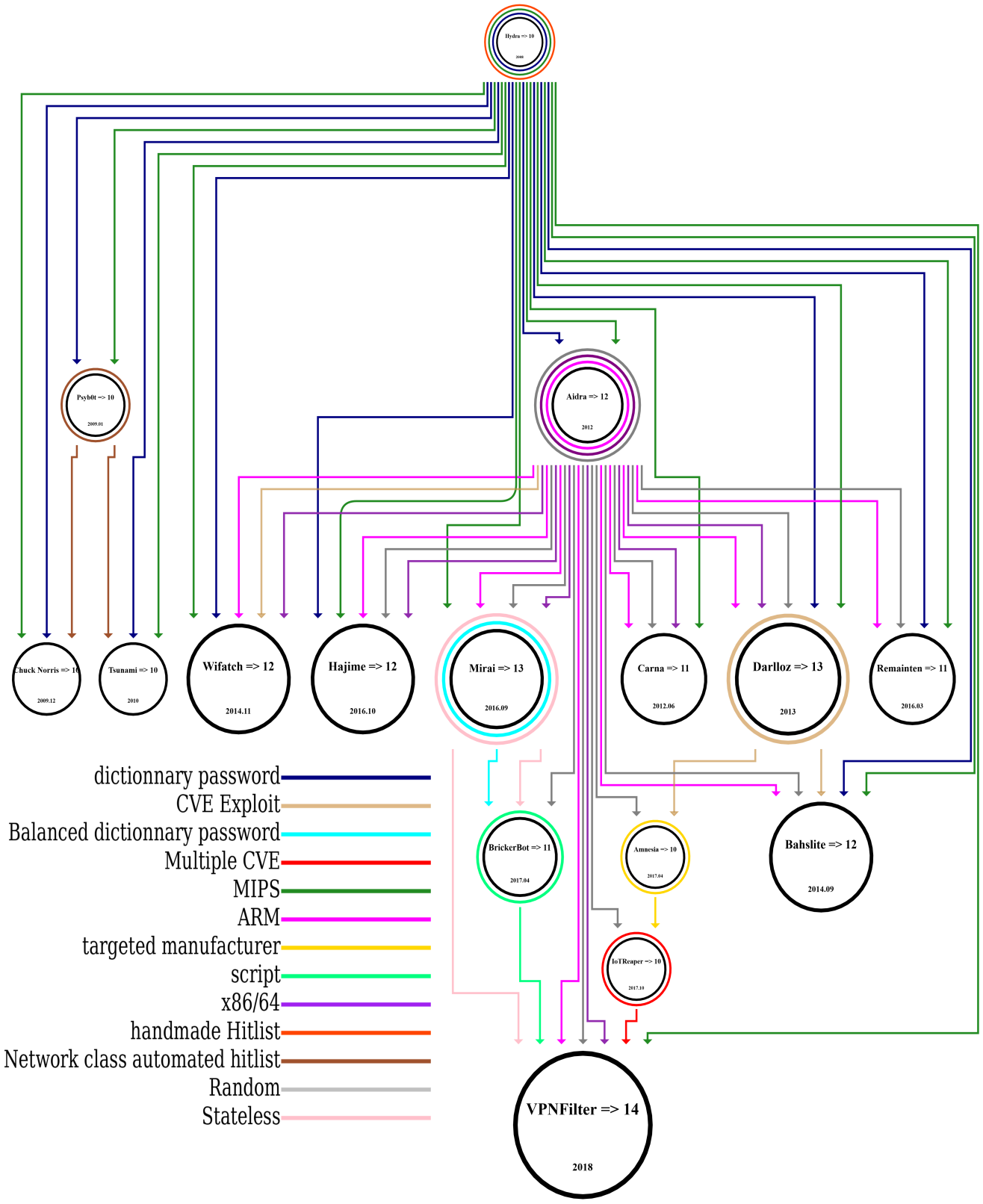

Figure 3.4 – Propagation des fonctionnalités d'infection.

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.