Systèmes coopératifs décentralisés de détection et de contre-mesures des incidents et attaques sur les réseaux IP

Hachem Guerid

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2014-ENST-0079

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et réseaux »

présentée et soutenue publiquement par

Hachem GUERID

6 Decembre 2014

Systèmes coopératifs décentralisés de détection et de contre-mesures des incidents et attaques sur les réseaux IP

Directeur de thèse : **Ahmed SERHROUCHNI**

Jury

Mme Dominique GAÏTI, Professeur, Université de Technologie de Troyes	Rapporteur
M. Omar CHERKAOUI, Professeur, Université du Québec à Montreal	Rapporteur
M. Abdelmadjid BOUABDALLAH, Professeur, Université de Technologie de Compiègne	Examinateur
M. Rida KHATOUN, Maître de Conférences, Télécom ParisTech	Examinateur
M. Karel MITTIG, Ingénieur de Recherche, Orange Labs	Examinateur

TELECOM ParisTech
École de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

À la mémoire de mon père

Remerciements

Je tiens à remercier les membres du jury qui ont accepté d'évaluer mon travail. Je remercie les rapporteurs Madame Dominique Gaïti, Professeur à l'Université de Technologie de Troyes et Monsieur Omar CHERKAOUI, Professeur à l'Université du Québec à Montréal. Je suis reconnaissant pour l'intérêt qu'ils ont porté à mes travaux de recherche, tout en ayant un regard critique et juste. Je remercie également Monsieur Abdelmadjid BOUABDALLAH, Professeur à l'Université de Technologie de Compiègne et Monsieur Rida KHATOUN, maître de Conférences à Télécom ParisTech d'avoir accepté de faire partie de mon jury de thèse et pour leurs questions pertinentes pendant la soutenance.

Je remercie très chaleureusement Karel Mittig pour son encadrement au sein d'Orange Labs et pour son amitié pendant ces années. Je lui exprime ma sincère reconnaissance pour sa disponibilité, son suivi continu de mes travaux, et pour ses qualités scientifiques et humaines.

Je remercie très vivement mon directeur de thèse le Professeur Ahmed Serhrouchni d'avoir accepté de me diriger patiemment pendant ma thèse de doctorat. Je lui exprime toute ma gratitude et ma reconnaissance pour ses encouragements, ses conseils, sa rigueur qui ont grandement contribué à l'aboutissement de cette thèse

Je n'oublie pas mes collègues d'Orange Labs et de Télécom ParisTech, et plus particulièrement Abdelhamid, Brahim, Chrystel, Kahina, Mohammed, Saïd, et Youcef pour leurs conseils et leur soutien au quotidien.

Ce travail n'aurait pas pu être mené à son terme sans le soutien et la patience de mes amis, Saïd, Mohamed, Ilies et Abdou. Une dédicace très particulière à Majid qui m'a supporté pendant les périodes difficiles.

Finalement, je souhaite remercier ma famille, ma mère, mes soeurs Fadia et Lamia, pour leur amour, leur sacrifice et leur soutien permanent pendant cette thèse et durant l'ensemble de mes études.

Résumé

Résumé

La problématique des botnets, réseaux de machines infectées par des logiciels malveillants permettant de les contrôler à distance, constitue une préoccupation majeure du fait du nombre de machines infectées et des menaces associées : attaque par déni de service distribué (DDoS), spam, vol de données bancaires. Les solutions de lutte contre les botnets proposées présentent des limitations majeures dans le contexte d'un opérateur réseau (contraintes de volumétrie et de passage à l'échelle, respect de la confidentialité et de la vie privée des utilisateurs).

Cette thèse propose quatre contributions orientées réseau de lutte contre les botnets. Chaque contribution traite d'une étape complémentaire dans la problématique des botnets : la première contribution permet de remonter à la source d'attaques par déni de service, et ainsi d'identifier un groupe de machines infectées à l'origine de ces attaques. La deuxième contribution concerne la détection des communications entre les machines infectées et leurs serveurs de contrôle et commande dans un réseau à large échelle, et offre ainsi l'opportunité de bloquer ces serveurs pour limiter le risque de nouvelles attaques. La troisième contribution permet une détection collaborative de botnets dans un contexte inter-domaine et inter-opérateur, permettant ainsi de lutter contre l'aspect hautement distribué de ces botnets. Enfin, la dernière contribution proposée permet de remédier aux botnets en ralentissant les communications entre les machines infectées et leur serveur de contrôle, offrant par ce biais une contre-mesure aux stratégies d'évasions développées par les cybercriminels afin de rendre leurs botnets plus résilients.

Collaborative and decentralized detection and mitigation of network attacks

Abstract

The problem of botnets, networks of infected hosts controlled remotely by attackers, is a major concern because of the number of infected hosts and associated threats, like distributed denial of service (DDoS), spams, and data theft. State of the art solutions to

fight against botnets have major limitations in a context of a network operator (scalability of the solution, confidentiality and privacy of users).

In this thesis, we propose four network-based contributions to fight against botnets. Each solution address a different and complementary issue in this area: the first contribution tracebacks the source of denial of service attacks which threaten the network availability, allowing by that way to identify infected devices used to perpetrate these attacks. The second contribution detects the communications between infected computers and their command and control server (C&C) in a large scale network and offers the opportunity to block these servers to minimize the risk of future attacks. The third contribution enables collaborative detection of botnets in an inter-domain and inter-operator context in order to fight against the highly distributed aspect of these botnets. Finally, the last contribution mitigates botnets by slowing down the communication between infected hosts and their C&C server, providing a countermeasure against evasion techniques developed by cybercriminals to make their botnets more resilient.

Table des matières

Introduction

Les machines connectées à internet sont la cible d'attaques variées afin de leur subtiliser des données ou de tirer profit de leurs ressources. Ces attaques sont menées essentiellement par des botnets, réseaux de machines infectées contrôlées à distance de manière synchronisée par un attaquant appelé l'opérateur du botnet. Elles sont contrôlées en général au moyen d'un serveur de contrôle baptisé serveur de contrôle et commande du botnet (C&C).

Les botnets représentent actuellement la principale source des attaques réseau telles que le spam, les dénis de service distribués (DDoS), la fraude aux clics et le vol de données. En effet, les botnets sont à l'origine de 88% des spams, eux-mêmes représentant 80% du trafic mail total [SSPS12], et les sommes détournées par la fraude au clic représentent 19.1% du chiffre d'affaires de la publicité en ligne [CdQC12].

Pour contrôler leurs botnets, les attaquants doivent communiquer régulièrement avec les machines infectées, ce qui génère du trafic réseau. De plus, dans une structure centralisée, les machines infectées se connectent vers un même serveur de contrôle. Ainsi, une approche par analyse du trafic réseau s'avère propice pour la détection de botnets. En outre, une approche réseau de détection de botnet est complémentaire aux solutions d'antivirus installées au niveau des hôtes, et permet de pallier les limitations de ces derniers.

Afin d'ajouter plus d'agilité à leur architecture et d'éviter de se faire détecter, les cybercriminels s'appuient sur des protocoles et des services usuels pour communiquer avec leurs machines infectées, tel que les protocoles HTTP, IRC et DNS.

L'augmentation du débit des communications ces dernières années et la ressemblance du trafic généré par les activités légitimes et les activités malveillantes rends difficile l'identification de ces dernières. En effet, pour recevoir leurs commandes d'attaques, les membres d'un botnet de type HTTP se comportent généralement comme les visiteurs d'un site internet légitime. Ceci complique leur détection au niveau réseau, particulièrement au niveau d'un opérateur réseau vu le volume de trafic basé sur ce protocole.

Pour lutter contre les botnets, des méthodes de blocage basées sur des listes d'adresses de serveurs de contrôle de botnets ont été déployées par les réseaux d'opérateurs et les réseaux d'entreprise. Cependant, afin de contourner ce blocage, les concepteurs de botnets ont développé des techniques calquées sur les approches de haute disponibilité utilisées dans les architectures Internet et réseau. En effet, ces techniques leur permettent ainsi de changer fréquemment de point de ralliement pour leur serveur de contrôle.

Les botnets ne connaissent pas les frontières nationales. Leurs machines infectées sont distribuées dans des pays et des opérateurs différents. Ceci induit une distribution des attaques qui nécessite une collaboration des autorités des pays et opérateurs considérés pour y remédier, avec toutes les contraintes juridiques et réglementaires que cela implique.

Le travail mené dans cette thèse se positionne sur ce contexte, et vise à répondre à

cette problématique des botnets en intervenant sur les différentes étapes de leur cycle de vie.

Afin d'illustrer nos différentes contributions, nous nous sommes ainsi basés sur le scénario d'une attaque par déni de service distribué (DDoS) sur une infrastructure réseau. Cette attaque vise à saturer les liens réseau de la cible jusqu'à ce qu'elle devienne incapable d'offrir des services à ses utilisateurs légitimes. De plus, cette attaque peut également impacter son opérateur réseau puisque cela lui induit une surcharge réseau sur l'ensemble des liens réseau menant à cette cible.

Les attaques par déni de services distribuées (DDoS) sont menées par des machines infectées, contrôlées généralement par l'opérateur du botnet au moyen d'un serveur de C&C. La distribution de ces machines et l'addition de leurs débits respectifs rendent l'impact de cette attaque particulièrement important. De plus, les machines infectées peuvent masquer leurs adresses à la cible. Ainsi, cette dernière ne sera pas en mesure de se baser sur leur localisation pour mettre en place des contre-mesures appropriées.

Afin de permettre à la cible de remonter aux sources de ces attaques et d'appliquer des méthodes de remédiation, nous proposons une méthode de traçabilité du trafic IP. Notre méthode permet de remonter à la source d'attaques par déni de service directes et indirectes basées sur le protocole ICMP, comme l'attaque SMURF.

Les sources de ces attaques correspondent à des machines contrôlées par un botnet. Elles reçoivent les commandes d'attaques par l'intermédiaire d'un serveur de C&C au moyen d'un canal de contrôle. Ainsi, le démantèlement d'un botnet implique l'identification au préalable de ce canal et sa déconnexion afin d'empêcher les commandes de contrôle d'atteindre les machines infectées.

Pour identifier ce canal, nous proposons une méthode de détection de botnets dans un réseau à large échelle. Notre méthode a pour objectif de détecter les serveurs de contrôle de botnets qui utilisent la technique domain-flux pour contourner le blocage de ces serveurs. Notre méthode se base uniquement sur le trafic généré par les machines infectées lors de leurs tentatives de rejoindre le serveur de contrôle du botnet. Se baser sur ces communications permet de détecter les botnets, quelles que soient les attaques qu'ils génèrent.

Notre méthode de détection de botnets se base sur l'analyse du trafic d'un opérateur réseau à large échelle, c'est-à-dire qui est généré par plusieurs millions de clients. Ainsi, elle prend en compte les contraintes de volumétrie et de préservation de la vie privée des utilisateurs de ce réseau.

Puisque les membres d'un botnet sont distribués entre plusieurs opérateurs réseau, et que ces derniers ne peuvent pas partager le trafic réseau de leurs clients, l'analyse du trafic de ces opérateurs en vue de la détection de botnet ne peut pas être faite de manière centralisée. Pour remédier à cela, nous proposons une méthode collaborative et inter-opérateur de détection de botnet. Notre méthode de détection se base sur la coordination et la ressemblance du comportement et des attaques générées pour identifier les membres d'un même botnet. De plus, notre méthode prend en compte les contraintes légales et opérationnelles lors du partage d'informations entre les différents domaines réseau en vue de la détection.

Pour que les machines infectées ne soient plus contrôlées par l'attaquant, il faut les empêcher de se connecter à leur serveur une fois que ce dernier a été identifié. Cependant, les attaquants se sont adaptés à la lutte des botnets, et ils ont mis en oeuvre des techniques pour contourner le blocage de leur serveur. De plus, puisqu'ils utilisent des canaux de contrôle chiffrés, les machines infectées peuvent identifier les tentatives d'usurpation de leur serveur de contrôle.

Pour pallier à cela, nous proposons une méthode de remédiation contre les botnets qui permet de ralentir les communications entre les machines infectées et leur serveur de contrôle. Le ralentissement des connexions permet d'empêcher les logiciels malveillants de se rendre compte de la remédiation et d'utiliser des serveurs ou des protocoles alternatifs.

Notre méthode de remédiation est suffisamment générique pour être appliquée à n'importe quel botnet. En effet, elle peut être appliquée à des logiciels malveillants sans connaitre au préalable leur méthode de chiffrement ou la sémantique des messages échangés.

Le mémoire est organisé comme suit :

- Le premier chapitre est consacré à l'analyse des botnets. Ainsi, nous étudions leurs différentes architectures, leur cycle de vie, leurs méthodes de communications, et les différentes attaques générées.
- Le deuxième chapitre est consacré à la traçabilité du trafic internet comme moyen d'identifier les machines infectées à l'origine d'une attaque. Après avoir analysé les approches de traçabilité proposées dans l'état de l'art, nous définissons notre méthode de traçabilité de trafic IP qui permet de remonter à la source d'attaques par déni de service directes et indirectes basée sur le protocole ICMP.
- Le troisième chapitre est consacré à la détection proactive de botnet dans un réseau à large échelle, c'est-à-dire avant qu'une attaque ne soit lancée par les machines infectées. Nous présentons notre nouvelle méthode de détection de botnets de type domain-flux qui permet une analyse en temps réel du trafic d'un opérateur réseau.
- Le quatrième chapitre est consacré à la détection collaborative de botnets. Nous présentons notre méthode collaborative et inter-domaine de détection de botnets.
- Le cinquième chapitre est consacré à la remédiation de botnets. Nous analysons dans un premier temps les différentes méthodes de réactions employées pour lutter contre les botnets et leur contournement par ces derniers. Ensuite, nous présentons notre méthode de remédiation qui permet de bloquer les logiciels malveillants en les empêchant de se connecter à leur serveur de contrôle.
- Enfin, dans le dernier chapitre, nous concluons ce mémoire par un récapitulatif des résultats obtenus et les perspectives pour de futurs travaux .

Chapitre 1

Analyse de botnets

1.1 Architecture, fonctions et cycle de vie

Les botnets, réseaux de machines infectées [CJM05] par des logiciels malveillants permettant de les contrôler à distance, constituent une préoccupation majeure pour les opérateurs réseau du fait du nombre de machines infectées et des menaces associées : attaques par déni de service distribuées (DDoS), spam, et vol de données bancaires.

Ces machines infectées sont sous le contrôle de machines maîtres, qui sont à leur tour sous le contrôle d'attaquants. Les machines maîtres envoient des commandes vers les machines infectées afin de récupérer des informations volées au préalable ou pour déclencher d'autres attaques. La communication au coeur du botnet est assurée par des protocoles largement utilisés pour des activités légitimes comme HTTP et IRC, afin d'éviter d'être identifiés par les systèmes de détection réseau. Ceci en réutilisant des composants existants comme les serveurs et clients IRC et les serveurs web.

La communication des machines infectées, ou bots, avec les machines maîtres, ou serveurs de contrôle et commande (C&C), est soit centralisée, soit pair-à-pair (p2p). Dans une structure centralisée, les machines infectées communiquent directement avec les machines maîtres, contrairement à la structure décentralisée où une communication doit traverser plusieurs machines infectées avant d'atteindre la machine maître.

Le développement des moyens de paiement électroniques et la puissance de calcul des ordinateurs actuels, ainsi que l'augmentation de la bande passante réseau chez les particuliers, a attisé la convoitise des attaquants. En effet, le développement des botnets est devenu une activité commerciale, voire militaire [KWU12].

Afin d'exploiter au mieux les ressources des machines corrompues, les attaquants informatiques ont automatisé leurs procédés pour les contrôler et déclencher les attaques de manière synchronisée. Cette automatisation correspond à la notion de botnets où son opérateur (botmaster) contrôle les ressources des machines infectées et décide du déclenchement d'attaques distribuées.

La motivation principale des créateurs de botnets est d'engranger des profits en utilisant les ressources et les données des machines sous leur contrôle. Ils font des profits en vendant les données subtilisées aux victimes ou en exploitant les ressources réseau de leurs machines pour déclencher des attaques. Par exemple dans l'attaque spam, les opérateurs de botnets sont rémunérés par des tiers pour l'envoi massif de courriers indésirables.

Le développement des moyens de paiement électronique a permis aux botnets de générer des profits de manière plus directe. Les botnets bancaires sont apparus. Ils permettent l'interception de données telles que les numéros de carte de crédit, ou les

identifiants de connexion vers le compte bancaire. Ces données sont revendues ensuite vers des tiers.

L'évolution majeure des botnets est l'apparition de kits de création de botnets, communément appelés Crimeware. Ces derniers permettent la mise en place des différents composants d'un botnet : le logiciel malveillant à installer sur la machine de la victime ainsi que le serveur de contrôle et commande. Ces kits sont vendus à des prix attractifs qui varient selon les fonctionnalités du botnet créé. Par exemple, le Crimeware Beta bot est vendu entre 300 et 500 $ [bet13], alors que le botnet Aldibot [ald11] est vendu à moins de 7 $.

La distribution de ces kits à des prix attractifs a permis la "démocratisation" des botnets. Ainsi, des individus qui n'ont pas de compétences techniques élevées peuvent acquérir ces kits pour créer leur propre botnet et le contrôler sans difficulté. Pour que cette activité soit lucrative, le botnet contrôlé doit contenir des machines infectées. Autrement dit, le logiciel malveillant doit être exécuté par des machines pour qu'elles rejoignent le botnet. Les opérateurs de botnets doivent infecter de nouvelles machines ou passer par des tiers le faire.

À titre d'exemple, le kit de création de botnets Zeus, très utilisé pour le vol de données bancaires, contient un logiciel permettant la génération du programme malveillant du botnet, ainsi qu'un ensemble de pages web. Ces pages doivent être hébergées dans un serveur web pour permettre à l'opérateur de contrôler son botnet. Ce kit était vendu entre 3000 et 4000 $ [SJ11] lors de son apparition en 2007.

1.1.1 Types de structure de botnets

Un botnet peut être organisé suivant une structure centralisée ou une structure décentralisée comme illustrée dans la figure 1.1. Ces 2 architectures peuvent parfois être combinées pour former des structures hybrides. De cette structure dépend le type de communication entre les machines infectées et leur serveur de contrôle et commande (C&C). Le choix du type de structure du botnet impacte largement son fonctionnement, sa résilience et sa réactivité.

Structure centralisée

Dans ce type de structures, le botnet est contrôlé par un ou plusieurs serveurs de contrôle et commande. Les membres du botnet se connectent directement au serveur de contrôle et commande afin de recevoir leurs instructions ou pour renvoyer les rapports des attaques accomplies. Ce serveur est contrôlé par l'opérateur du botnet. Les commandes d'attaques sont reçues directement, ce qui permet à l'opérateur de déclencher les attaques rapidement.

Dans les structures centralisées, le serveur de contrôle et commande est un point de défaillance du botnet. Une panne de ce serveur ou l'arrêt des connexions vers celui-ci aura pour conséquence l'interruption du botnet. L'opérateur de ce botnet ne le contrôlera plus puisqu'il ne peut plus envoyer de commande vers les machines infectées.

Lorsque les botnets se basent sur des protocoles populaires pour contrôler leurs machines infectées, ils deviennent difficilement détectables au niveau réseau, puisque ces protocoles sont également utilisés par les applications légitimes. Ainsi, les systèmes de détection d'intrusion (IDS) qui se basent sur les signatures des communications malveillantes peuvent difficilement et au prix de beaucoup de faux positifs différencier une

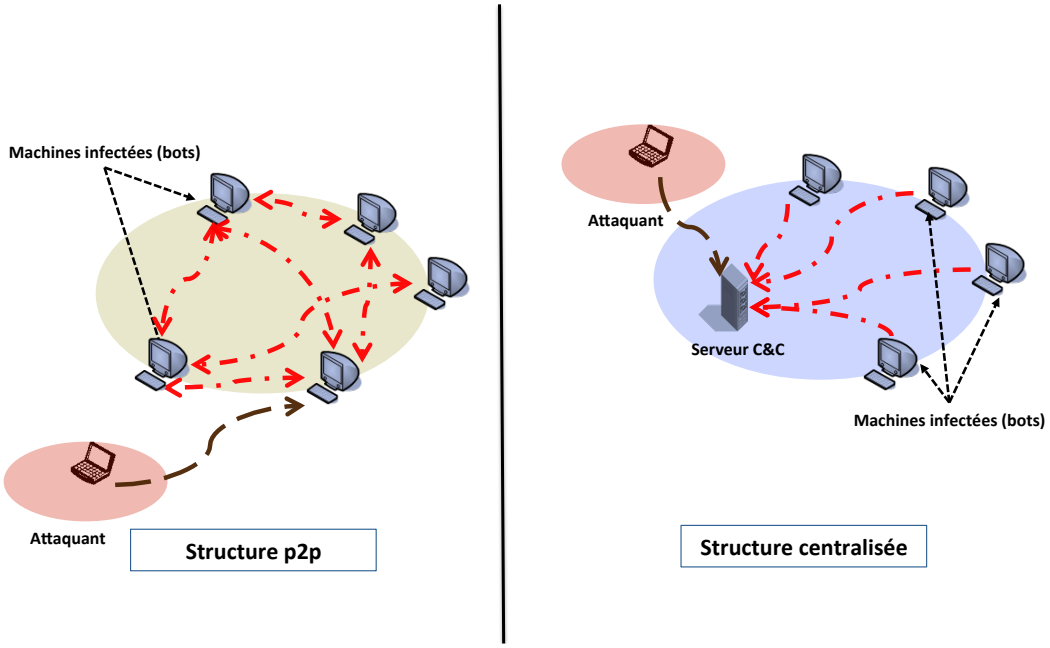

FIGURE 1.1 – Structures de botnets : centralisée et pair-à-pair

communication légitime d'une communication malveillante d'un botnet si cette dernière se base sur les mêmes protocoles.

Dans une structure centralisée, le serveur de contrôle doit pouvoir gérer l'ensemble des connexions des machines infectées de ce botnet. Ce serveur peut ne pas supporter la charge lorsque le botnet comporte un nombre important de machines à contrôler. Afin de remédier à cela, les opérateurs de botnets font de l'équilibrage de charge en utilisant plusieurs serveurs. En outre, l'utilisation de plusieurs serveurs de contrôle rend le démantèlement du botnet difficile. En effet, il faut déconnecter l'ensemble des serveurs afin que l'opération soit réussie.

Pour contrecarrer les opérations menées contre les botnets, les attaquants se basent sur des techniques pour cacher l'emplacement du serveur de contrôle du botnet. Ceci a pour conséquence de rendre ces opérations onéreuses puisqu'il leur faut prévoir l'adresse du serveur de contrôle pour pouvoir appliquer des contre-mesures.

Structure décentralisée

Dans une structure décentralisée ou pair-à-pair (P2P), la communication entre une machine infectée et l'attaquant s'effectue par l'intermédiaire des autres machines infectées du botnet. Pour propager les commandes dans le réseau, l'opérateur du botnet se connecte à un noeud du réseau.

Dans une structure décentralisée, contrairement à la structure centralisée, il n'y a pas de point de défaillance dans le réseau. La perturbation de ce type de structure est difficile, puisque même si une partie des machines infectées du botnet n'est pas joignable, l'autre partie reste toujours fonctionnelle. En effet, l'attaquant peut contrôler cette partie du botnet en se connectant à n'importe quel noeud de cette partie.

Cependant, ce type de botnet est vulnérable aux infiltrations. Ainsi, le botnet décentralisé Storm [HSD+08] a été infiltré par des chercheurs [LWM09]. Dans un premier temps, ils ont récupéré la clé de chiffrement utilisée dans le canal de contrôle pour analyser

les commandes envoyées par l'attaquant. Ensuite, en se faisant passer par l'opérateur du botnet, ils ont envoyé des commandes auprès des machines infectées, afin de leur demander de désinstaller le logiciel malveillant.

Bien que l'opération d'infiltration de ce type de botnet peut être réussie, elle n'est réservée que pour lutter contre les botnets représentant une menace importante que ce soit par le nombre important de leurs machines ou par la nuisance de leurs attaques. En effet, la récupération de la clé de chiffrement du botnet pour analyser les communications et pour pouvoir reproduire les commandes est couteuse puisqu'il faut faire de l'ingénierie inverse sur le fichier binaire du logiciel malveillant du botnet.

Puisqu'il n'y a pas d'entité centralisée, les machines infectées dans ce type de botnets doivent se connecter aux autres membres du réseau pair-à-pair pour rejoindre le botnet. Pour cela, ils se connectent à des points d'entrée appelés "supernodes". Ces derniers font partie du botnet et procurent aux machines nouvellement infectées les adresses des autres membres. Cette opération est appelée ralliement dans les réseaux p2p.

Les supernodes doivent être accessibles en permanence puisque le ralliement de nouveaux membres au botnet dépend de leur accessibilité. Ainsi, les machines nouvellement infectées doivent avoir en leur possession les adresses IP ou les noms de domaine des supernodes. Autant dire que ces adresses ou noms de domaine doivent être configurés dans le logiciel malveillant du botnet. Par conséquent, si ces supernodes ne sont pas accessibles, le botnet ne peut pas accueillir de nouveaux membres.

Les commandes émises par l'attaquant dans une structure décentralisée mettent plus de temps à atteindre les machines infectées par rapport à une structure centralisée, puisqu'elles doivent traverser plusieurs noeuds avant d'attendre leurs destinations. Dans cette structure, l'opérateur du botnet contrôle conjointement les machines infectées. Les commandes qu'il génère sont destinées à l'ensemble des machines infectées, contrairement à la structure centralisée où il est plus facile de contrôler individuellement les machines infectées. De plus, les communications de ces botnets sont plus faciles à détecter dans certains contextes comme dans les réseaux d'entreprise où le P2P est interdit.

Structure hybride

Puisqu'il est aisé de bloquer les communications vers un serveur, les botnets centralisés sont sensibles aux défaillances et à la perturbation de leurs opérations. Le réseau local ou le réseau de l'opérateur des machines infectées peut bloquer les communications vers leurs serveurs de C&C. Afin de contourner ce blocage potentiel, des opérateurs de botnet utilisent une structure hybride.

Cette structure hybride fonctionne comme une structure centralisée si aucune tentative de blocage ou de perturbation du botnet n'est en cours. En revanche, si les serveurs de contrôle ne sont plus joignables par les machines infectées, le botnet bascule dans un mode pair-à-pair. Les machines infectées se connectent les unes aux autres afin de créer un réseau pair-à-pair qui pourra être accessible par l'opérateur du botnet. Le botnet peut également changer de structure périodiquement pour diminuer les chances de se faire détecter ou pour renforcer la structure de son réseau pair-à-pair.

Cette structure hybride a été adoptée par des botnets connus comme Zeus [Zeub].

1.1.2 Cycle de vie

Pour qu'un botnet soit profitable à son opérateur, il doit contenir des machines infectées à contrôler qui exécutent le logiciel malveillant de ce botnet. Ce logiciel permet

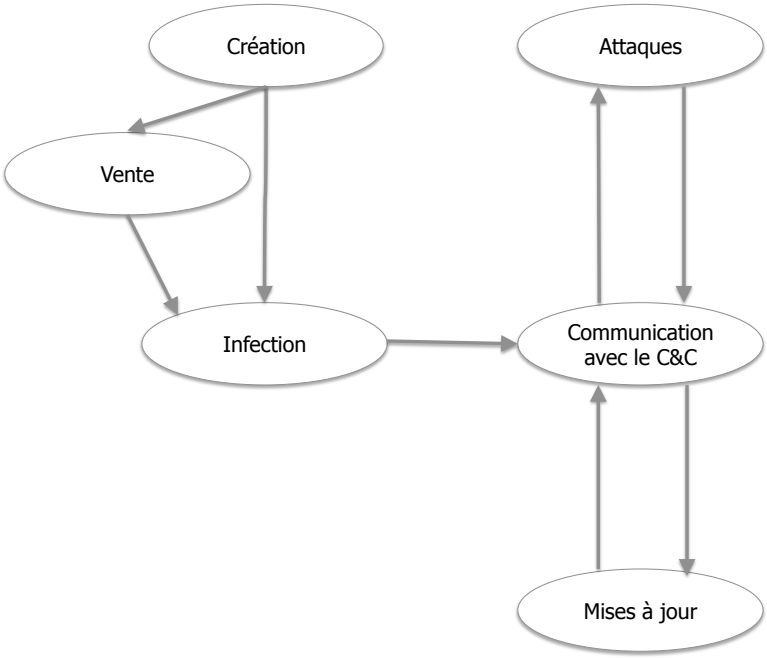

FIGURE 1.2 – Les différentes étapes du cycle de vie d'un botnet

de corrompre et de passer la machine qui l'exécute sous le contrôle de l'attaquant pour déclencher des attaques ou pour intercepter les données de la victime.

Les machines infectées sont connectées en permanence avec le C&C du botnet, afin de recevoir les commandes d'attaques, et pour envoyer les données dérobées et les comptes-rendus des attaques accomplies. De plus, l'opérateur du botnet met à jour en permanence son réseau et son logiciel malveillant, afin d'ajouter de nouvelles fonctionnalités à son botnet ou pour contrecarrer les opérations de remédiation.

Le cycle de vie ordinaire d'un botnet est constitué de : la création, l'infection, la communication avec le serveur de C&C, les attaques, et la mise à jour. Ceci est illustré dans la figure 1.2.

La Création

L'histoire des botnets remonte à Eggdrop, en 1993 [egg]. Ce bot n'avait pas d'intention malveillante. Il a été créé afin d'aider à la gestion des canaux de discussion dans les serveurs IRC [OR93]. Les bots Eggdrop reçoivent et interprètent des commandes afin de fournir des services aux utilisateurs du canal.

Plusieurs botnets basés sur le protocole IRC sont apparus à la fin des années 1990. Toutefois, le tournant majeur dans le développement des botnets est la diffusion du code source de logiciels malveillants tel que Agobot, SDBot, et GT-Bot [ZTKM11]. La particularité du code source de ces logiciels est qu'il était possible et sans grande difficulté d'ajouter de nouvelles fonctionnalités au botnet initial[Hol05].

Les attaquants se sont basés sur ces codes sources afin de développer de nouveaux botnets. Ainsi, des milliers de variantes du botnet Agobot ont été détectées par les solutions d'antivirus moins d'une année après la diffusion de son code source. De plus, les créateurs de botnets de type IRC se sont basés directement sur le code des clients et

des serveurs IRC légitimes.

Avec la popularité du web, les attaquants se sont tournés vers le protocole HTTP pour le canal de contrôle du botnet. Ils se sont basés sur l'architecture des serveurs web pour leur serveur de contrôle et commande C&C. L'utilisation des serveurs web leur permet d'assimiler leur trafic de contrôle au trafic légitime et de ne pas attirer l'attention des systèmes de détection. En effet, ils génèrent le même type de trafic réseau que les utilisateurs légitimes.

Une autre étape importante dans le développement des botnets est la divulgation du code source du botnet bancaire Zeus en mai 2011 [zeuc]. Comme la divulgation du code source de botnets IRC a engendré plusieurs variantes, plusieurs botnets se sont basés sur son code source telles que Citadel [RZPD14] et GameOver [ARSG+13]. La divulgation du botnet Carberp [Car], plus évolué que Zeus, ferra vraisemblablement l'objet de plusieurs variantes dans le futur.

Les développeurs de botnets sont de moins en moins impliqués dans la gestion journalière des machines infectées. Ils développent leur botnet avec un haut niveau d'abstraction, afin de pouvoir revendre leur code à un nombre important de clients. En effet, un même code peut engendrer un nombre important de botnets avec des ensembles de machines infectées disjoints et des serveurs de contrôle différents.

La Vente

L'émergence des kits de création de botnets prêts à l'emploi a introduit l'étape de vente dans le cycle de vie d'un botnet. Ainsi, le premier kit de création de botnets de type Zeus [BOB+10] était vendu à plusieurs milliers de dollars [SJ11] à la fin des années 2000. Pour ce prix, l'acquéreur disposait d'un générateur permettant de créer des variantes du logiciel malveillant servant à infecter les machines, et un site web configurable permettant de contrôler le botnet.

Afin d'empêcher la génération frauduleuse du botnet, les développeurs des kits de création de botnets ont intégré des mécanismes pour empêcher la distribution ou la revente de leurs kits. Les développeurs du botnet Zeus ont mis en oeuvre le même procédé pour protéger leur produit que celui utilisé pour l'activation des produits Microsoft. Ainsi, l'acquéreur de ce kit doit fournir au préalable la configuration matérielle et logicielle de sa machine au vendeur [zeua]. Par conséquent, il est le seul à pouvoir utiliser le kit et créer un botnet.

Après l'apparition de Zeus, plusieurs autres kits de création de botnets sont apparus, dont le plus connu est SpyEye, concurrent direct de Zeus. Il intégrait dans son code une fonctionnalité [Coo10] pour supprimer Zeus de la machine de la victime avant de s'installer lui-même.

En parallèle des kits de création de botnets destinés à un large public, certains sont développés sur mesure pour certains clients. Par exemple, le botnet Mariposa a été développé par un Slovène, mais était opéré par des Espagnols [Gol11]. L'avantage de ne pas passer par des kits de création de botnets est d'éviter de partager des signatures, sur le logiciel malveillant ou sur le trafic réseau, avec les autres botnets créés avec le même kit. En effet, le canal de contrôle du botnet Mariposa était basé sur un protocole réseau propriétaire [SBBD10].

L'infection

Pour avoir des machines sous leur contrôle, les opérateurs de botnets doivent exécuter leur logiciel malveillant sur les machines des victimes. Ce logiciel malveillant permet à l'opérateur du botnet de prendre le contrôle de la machine, autrement dit, de l'infecter. Ainsi, cette dernière rejoint le botnet et attend la réception de commandes d'attaque.

L'infection d'une machine peut s'effectuer de manière directe ou indirecte. Dans la méthode indirecte, un programme d'infection achemine le logiciel malveillant et lance son exécution dans la machine de la victime [TWSO10]. Ce programme d'infection est appelé "dropper" puisqu'il permet de déposer le logiciel malveillant. Par contre, l'infection directe n'utilise pas de "dropper". La victime exécute directement le programme malveillant dans sa machine et rejoint le botnet.

Afin d'éviter que le programme malveillant ne soit détecté lors de l'infection par les solutions de sécurité qui se basent sur la signature des fichiers, les opérateurs de botnet chiffrent le contenu du fichier binaire de leur logiciel malveillant. Ceci permet d'empêcher les solutions d'antivirus et les systèmes d'intrusion de lire les instructions contenues, et ainsi d'inférer le comportement du logiciel malveillant. Les solutions de chiffrements les plus utilisées sont appelées "Packer" [YZA08]. Elles permettent de retarder le déchiffrement des instructions du programme malveillant jusqu'à son chargement dans la mémoire de la machine de la victime.

Une autre parade pour contourner les mécanismes de détection des programmes indésirables, consiste à signer ces programmes malveillants avec des certificats légitimes. En effet, des opérateurs de botnets ont signé leur programme malveillant avec des certificats de programmes légitimes volés [fak].

Les techniques d'infection varient considérablement. On décrit ci-dessous les principales techniques :

- **Vulnérabilité logicielle :** Lors de cette infection, la machine ciblée est introspectée à distance afin d'obtenir des informations sur les services en cours d'exécution, ainsi que les différents ports réseau ouverts. Si elle détecte un service vulnérable est en cours d'utilisation, le programme d'infection essaie de l'exploiter et d'exécuter le programme malveillant du botnet. .
- **Drive-by-Download :** le programme d'infection est téléchargé et exécuté par la victime lors de la consultation d'une page web compromise. Elle consulte cette page, soit parce que cette dernière est légitime, mais a été piratée, ou parce qu'elle a été redirigée vers cette page par le biais d'un spam ou d'ingénierie sociale.
  Le programme d'infection est téléchargé soit de manière automatique soit de manière manuelle. Il est téléchargé et exécuté automatiquement et sans informer la victime en exploitant une vulnérabilité dans le navigateur ou dans une des extensions installées. Le programme d'infection est téléchargé et exécuté manuellement par la victime parce que l'attaquant l'a fait passer pour un programme légitime. Par exemple, le programme d'infection peut se faire passer auprès de la victime pour un lecteur multimédia pour lire une vidéo proposée sur la page compromise.
  Afin d'automatiser le processus de recherche et d'exploitation des vulnérabilités des navigateurs des victimes potentielles, les attaquants utilisent des kits d'exploits tels que Blackhole ou Eleonore [GBC+12].
- **Infection par média externe :** Dans cette infection, le programme malveillant est transporté dans un média externe. Lorsque ce média est connecté à une machine, le programme malveillant est exécuté automatiquement et le système devient corrompu et rejoint le botnet. La particularité de cette infection est que les médias qui seront branchés à l'avenir au système corrompu transporteront à leur tour ce

programme malveillant et pourront infecter d'autres machines.
- **Cheval de Troie :** L'attaquant dissimule un code d'infection dans un programme ou un fichier légitime. Lorsqu'une machine exécute le programme légitime, le code d'infection est exécuté en parallèle. Par conséquent, le logiciel malveillant est installé et la machine rejoint le botnet . En effet, le programme d'infection du botnet CryptoLocker de type scareware [Ler13] est dissimulé dans des programmes légitimes partagés dans les réseaux d'échange de fichier pair-à-pair [cry].
- **Surinfection :** Certains attaquants se sont spécialisés dans l'infection des machines. Ils vendent les machines infectées aux opérateurs d'autres botnets. Ce service est appelé Pay Per Install (PPI) [CGKP11]. Le prix des machines infectées dépend de leur position géographique. Une machine dans un pays riche est vendue plus cher qu'une machine d'un pays en développement puisque l'attaquant peut espérer en retirer plus de profits.

Communication avec le serveur de C&C

Dès que la machine est infectée, elle essaye de communiquer avec son serveur de contrôle afin de le notifier de son infection et pour recevoir la dernière configuration. Les machines infectées sont en contact permanent avec leur serveur de C&C, afin de recevoir les dernières commandes d'attaque ou pour envoyer le résultat des attaques en cours. Le canal utilisé par les machines infectées pour communiquer avec leur serveur de contrôle est appelé canal de contrôle.

Attaques

Le but de l'attaquant disposant du contrôle d'un botnet est d'utiliser les machines infectées pour déclencher différentes attaques. Les machines infectées reçoivent les commandes d'attaques de la part du serveur de C&C, qui est contrôlé à son tour par l'attaquant. On peut regrouper les attaques générées par les botnets en deux catégories principales :
- **Attaques bruyantes :** Le trafic réseau généré lors de ces attaques peut facilement être identifié par un système d'analyse de trafic réseau. En effet, les machines infectées génèrent un trafic réseau malveillant important. On peut citer par exemple : le scan ou le balayage du réseau à la recherche de machines vulnérables, l'envoi massif de courriers indésirables, et la participation dans des attaques de déni de service.
- **Attaques furtives :** Contrairement aux attaques précédentes, les attaques furtives ne génèrent pas de trafic réseau important. Ainsi, ces attaques sont difficilement détectables par une solution d'analyse de trafic réseau. Elles surviennent essentiellement à l'intérieur du système de la victime. On peut citer par exemple : le vol de données des utilisateurs, et la fraude à la publicité en ligne.

Mise à jour

L'opérateur du botnet entretient son réseau en envoyant continuellement aux machines infectées des mises à jour du programme malveillant. Ces mises à jour peuvent contenir de nouvelles fonctionnalités, des corrections de vulnérabilités, ou une amélioration de la méthode de communication du botnet.

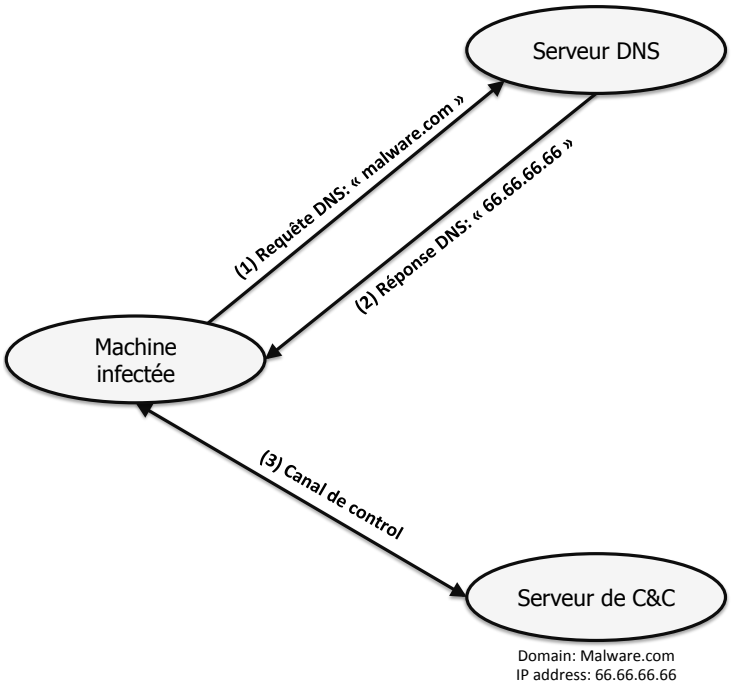

FIGURE 1.3 – établissement d'une connexion avec le serveur de C&C

1.2 Méthodes de communication

Les machines infectées se connectent régulièrement au serveur de contrôle et commande afin de recevoir des instructions ou pour envoyer des rapports sur les attaques en cours ou accomplies. Ils utilisent pour leurs communications des protocoles réseau usuels afin de se confondre avec le trafic légitime et ne pas attirer l'attention des systèmes de détection basés sur l'analyse du trafic réseau.

Pour atteindre le serveur de contrôle et commande du botnet, les machines infectées utilisent en majorité le protocole DNS pour récupérer son adresse IP. Ainsi, avant d'établir la connexion avec le serveur de contrôle, elles envoient des requêtes vers le serveur DNS comme illustré dans la figure 1.3. L'utilisation du protocole DNS permet d'ajouter plus de robustesse au botnet afin de contourner les dispositifs de blocage mis en oeuvre au niveau réseau.

Une fois l'adresse de leur C&C déterminée, les machines infectées se connectent à ce serveur en utilisant des protocoles usuels. Dans les premiers botnets, le protocole IRC (Internet Relay Chat) [OR] était utilisé pour monter le canal de contrôle. Cependant, la popularité du web et la facilité de la mise en oeuvre d'un serveur web ont poussé les concepteurs de botnets à utiliser le protocole HTTP.

Afin de gêner les méthodes de détection de botnets basées sur les signatures des messages échangés et pour ralentir leur analyse, des opérateurs de botnets chiffrent les commandes envoyées.

1.2.1 Détermination de l'adresse du serveur de C&C (Protocole DNS)

Historiquement, l'adresse IP du serveur de contrôle et commande du botnet était codée de manière statique dans le logiciel malveillant. Cette approche simpliste présentait

cependant plusieurs inconvénients. Ainsi, si le serveur de contrôle était en panne ou inaccessible à cause d'une opération de blocage, le botnet n'était plus opérable puisque les machines infectées ne pouvaient plus se connecter au serveur de contrôle. De plus, l'utilisation d'adresse IP statique ne permet par à l'opérateur du botnet de changer aisément l'emplacement du serveur de contrôle

Pour résoudre cette problématique, les créateurs de botnets se basent sur le protocole DNS, afin de permettre aux machines infectées de localiser leurs serveurs malveillants. En possédant le nom de leur serveur de contrôle, elles envoient des requêtes vers le serveur DNS afin de récupérer l'adresse IP de ce serveur.

L'utilisation de DNS permet à l'opérateur du botnet d'ajouter plus de résilience à son serveur de contrôle. En effet, il peut changer l'emplacement du serveur ou rajouter d'autres serveurs pour contrôler son botnet sans pour autant mettre à jour le logiciel malveillant. Pour cela, l'opérateur du botnet intervient uniquement au niveau du système DNS en modifiant l'association entre le nom de domaine du serveur et son adresse IP. Ainsi, la machine infectée continue de solliciter le serveur DNS pour résoudre le même nom de domaine du serveur de contrôle. Par contre, elle recevra en réponse des adresses IP différentes.

L'opérateur du botnet peut aussi utiliser le protocole DNS pour faire de l'équilibrage de charge [NH08] entre différents serveurs de contrôle. Ainsi, les machines infectées se partagent les différents serveurs de contrôle. Elles reçoivent les adresses IP de ces serveurs dans les réponses DNS. De plus, cette distribution de charge permet de gêner les tentatives de perturbation du botnet, puisqu'il faut bloquer l'accès à tous les serveurs de contrôle pour rendre le botnet inopérable à l'attaquant.

Les attaquants exploitent les propriétés du système DNS afin de mettre en oeuvre des stratégies d'évasion des contre-mesures mises en oeuvre contre les botnets. Ces stratégies ont pour objectif de cacher l'emplacement du serveur de contrôle et pour contourner le blocage du nom de domaine du botnet. Les techniques principales sont : le fast-flux et le domain-flux.

- Dans la technique fast-flux [NH08], l'attaquant essaie de cacher l'emplacement du serveur malveillant. Pour cela, il associe au nom de domaine du serveur malveillant des adresses IP qui changent de manière dynamique. Ces adresses correspondent à des machines qui jouent le rôle de proxy inversé vers le serveur malveillant.
  Les informations contenues dans les réponses DNS ont une durée de vie limitée définie par le champ TTL (Time To Live). Ainsi, l'association entre le nom de domaine et les adresses IP sont valides le temps de la durée de la valeur du TTL. L'attaquant dans la technique fast-flux, utilise une faible valeur du TTL pour obliger les machines infectées à fréquemment faire des requêtes DNS pour le nom de domaine malveillant. Ces machines recevront dans les réponses, les différentes adresses IP des proxys inversés associées au nom de domaine malveillant.
  Cette technique n'est pas uniquement utilisée pour cacher l'emplacement du serveur de contrôle et commande d'un botnet, elle est aussi utilisée pour cacher l'emplacement de serveurs web hébergeant des données ou des activités illégales (phishing, propagation d'exploits...). Elle est également utilisée par certains CDNs [HGRF08].
- La technique domain-flux [SGCC+09] permet à l'attaquant de contourner le blocage appliqué au niveau DNS. Dans cette technique, les machines infectées génèrent périodiquement et de manière algorithmique une liste de noms de domaine valide. L'attaquant connaissant le résultat de l'algorithme de génération, réserve au préalable un nom de domaine pour l'associer au serveur de contrôle du bot-

net. Puisque, les machines infectées ignorent quel nom de domaine a été réservé, elles envoient des requêtes DNS vers tous les noms de domaine jusqu'à ce qu'elles accèdent au serveur de contrôle.

1.2.2 Le canal de contrôle

Lorsque les machines infectées possèdent l'adresse IP de leur serveur de contrôle, elles établissent la connexion avec lui afin de recevoir des commandes ou pour envoyer des rapports d'attaques accomplies. Ce canal de communication est appelé le canal de contrôle du botnet. Nous allons détailler les principaux protocoles réseau utilisés pour le canal de contrôle : le protocole IRC et le protocole HTTP.

Protocole IRC

C'est un protocole de messagerie instantanée qui était très populaire à la fin des années 90, c'est-à-dire, en même temps que l'apparition des premiers botnets. Dans ces derniers, les machines infectées se connectent au serveur de messagerie IRC afin de recevoir les commandes envoyées par l'attaquant.

Le protocole IRC est basé sur des salons de discussion ou des canaux [OR]. Les utilisateurs doivent être connectés à un canal pour envoyer et recevoir des messages. Lorsqu'un message est envoyé sur le canal, il est reçu par l'ensemble des utilisateurs connectés. En conséquence, dans un botnet IRC, les messages envoyés par l'attaquant sur le canal de contrôle sont reçus par tous les membres du botnet.

Dans les botnets IRC, les machines infectées sont connectées en permanence sur le canal de contrôle du botnet. Ainsi, ils sont très réactifs. En effet, lorsque l'attaquant émet une commande sur le canal de contrôle, elle est reçue instantanément par tous les membres du botnet. En conséquence, les machines infectées du botnet peuvent enclencher les attaques rapidement.

Les chercheurs ont proposé plusieurs méthodes de détection de botnets [LXG+09] basés sur l'analyse du trafic IRC. Ces méthodes essayent d'identifier les communications du canal de contrôle parmi le trafic IRC légitime.

Contrairement aux protocoles HTTP ou HTTPS, le protocole IRC est bloqué par la majorité des pare-feux d'entreprises. Ainsi, les machines infectées au sein d'un réseau d'entreprise ne peuvent pas se connecter à leur serveur IRC de contrôle. Ceci implique que même si la machine est infectée par le logiciel malveillant du botnet, elle n'est pas contrôlée par l'opérateur botnet et elle ne peut pas participer aux attaques distribuées.

Par ailleurs, la quasi-disparition du protocole IRC [IRC] au profit des messageries instantanées comme Yahoo ou Msn a rendu ce type de botnet facilement détectable, conduisant les développeurs de botnets à migrer vers d'autres protocoles.

Protocole HTTP

Compte tenu de la popularité du web ces dernières années, et grâce à la facilité avec laquelle un serveur peut être mis en place, les développeurs de botnets ont délaissé le protocole IRC et sont passés au protocole HTTP. En 2011, 75% des botnets [PAG12] se basaient sur HTTP pour leur canal de contrôle.

Dans un botnet de type HTTP, les informations envoyées par une machine infectée à l'opérateur du botnet sont uniquement reçues par l'attaquant. Elles ne sont pas reçues par les autres machines infectées ou les autres visiteurs de la page web de contrôle,

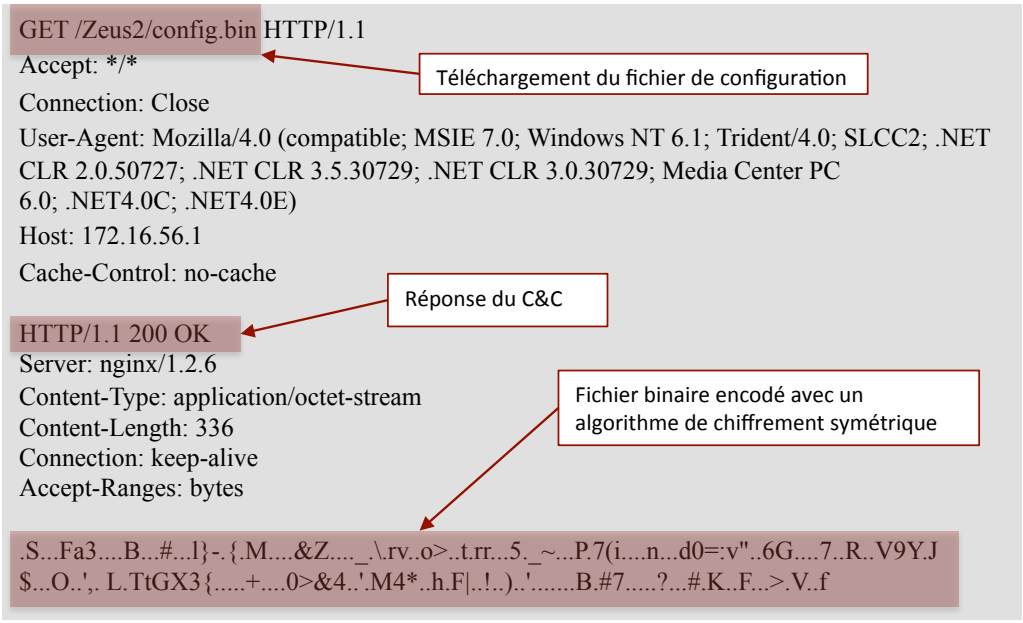

FIGURE 1.4 – Téléchargement du fichier de configuration (config.bin) du botnet Zeus

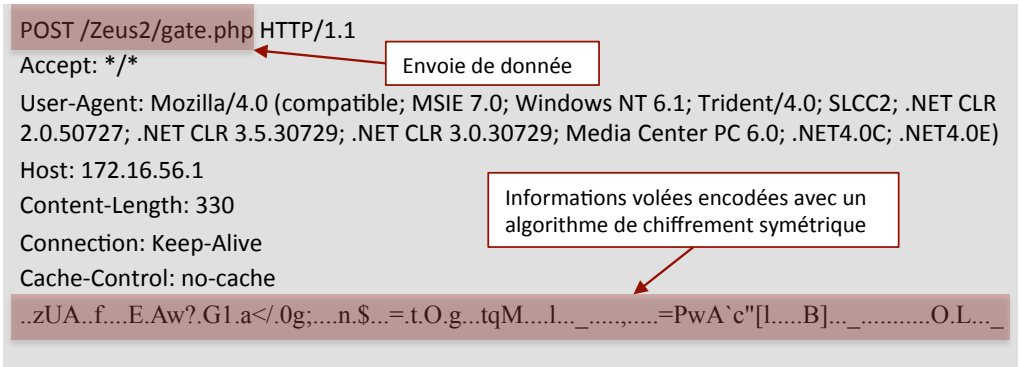

FIGURE 1.5 – Envoie d'informations volées au serveur de C&C du botnet Zeus

contrairement aux botnets de type IRC. Ainsi, dans un botnet de type HTTP, seul l'attaquant peut connaitre le nombre de machines infectées sous son contrôle.

La communication d'un botnet basé sur HTTP est basée sur le mode "PULL" puisque les machines infectées se connectent périodiquement au serveur web afin de retirer les dernières commandes émises par l'attaquant. Par contre, le protocole IRC est basé sur le mode "PUSH" puisque les machines infectées sont connectées en permanence au serveur de contrôle, donc elles reçoivent instantanément les commandes envoyées par l'opérateur du botnet. En conséquence, les botnets IRC sont plus réactifs par rapport aux botnets de type HTTP.

Dans le botnet HTTP Zeus, les machines infectées se connectent périodiquement à une adresse définie au préalable par l'opérateur du botnet afin de télécharger un fichier de configuration contenant les dernières commandes, comme illustré dans la figure 1.4. Les machines infectées se connectent également périodiquement vers une page du site de contrôle afin d'envoyer les dernières informations recueillies auprès de la victime comme illustrée dans la figure 1.5.

Le protocole HTTP est très populaire, que ce soit chez les particuliers ou chez les professionnels. Il représente la majorité du trafic internet en Europe [KDG+12a]. Il est rarement filtré par les pare-feux des entreprises. Ainsi, une machine appartenant à un réseau d'entreprise peut aisément appartenir à un botnet de type HTTP, contrairement aux botnets de type IRC.

Autres protocoles

Même si la majorité des botnets base leur canal de contrôle sur les protocoles HTTP et IRC, d'autres botnets utilisent des protocoles personnalisés. En effet, le botnet Mariposa [SBBD10] utilise un protocole au-dessus d'UDP appelé Iserdo Transport Protocol. Les botnets P2P se basent également sur des protocoles au-dessus d'UDP [GSN+07].

L'utilisation de protocoles personnalisés pour communiquer au sein d'un botnet permet d'échapper aux méthodes de détection basées sur des signatures spécifiques aux protocoles IRC et HTTP. L'inconvénient d'un protocole personnalisé est que les pare-feux réseau et particulièrement ceux d'entreprises peuvent bloquer les communications de ce protocole. En effet, les pare-feux peuvent être configurés pour bloquer les protocoles inconnus.

1.2.3 Chiffrement des communications

Les premiers botnets ne chiffraient pas leurs communications, et les messages du botnet circulaient donc en clair dans le réseau. Ces messages pouvaient être interceptés et analysés par des tiers. En conséquence, les solutions de sécurité pouvaient facilement identifier les communications malveillantes.

Plusieurs méthodes de détection de botnets se basent sur des signatures de séquences de communications malveillantes, que ce soit dans les commandes envoyées par le serveur de contrôle ou les rapports envoyés par les machines infectées. Ces signatures sont extraites à partir de l'analyse du trafic généré par des logiciels malveillants exécutés dans des environnements contrôlés.

Les communications en clair au sein d'un botnet permettent aux chercheurs en sécurité d'examiner les différents messages échangés. De fait, ils peuvent établir une communication avec le serveur de contrôle en se faisant passer pour une machine infectée, ou établir une communication avec une machine infectée en se faisant passer pour le

serveur de contrôle et commande. En effet, si les chercheurs peuvent usurper l'identité du serveur de contrôle, ils essaient d'envoyer des commandes aux membres du botnet pour désinstaller le logiciel malveillant.

Les développeurs de botnets ont commencé à chiffrer les communications entre les machines infectées et leur serveur pour empêcher ce type d'analyse. Les machines infectées et le serveur de contrôle partagent une clé de chiffrement. Ce chiffrement peut être appliqué à la totalité du canal de communication ou seulement sur les messages échangés. Ainsi, les botnets Zeus et Conficker [BOB+10, LW09] utilisent l'algorithme RC4 pour chiffrer les communications entre les machines infectées et le serveur de contrôle. De plus, le botnet Conficker utilise l'algorithme asymétrique RSA pour échanger les clés de chiffrement.

Les algorithmes de chiffrement symétrique tel que RC4 se basent sur une clé unique. Cette clé peut être codée dans le binaire du logiciel malveillant [BOB+10] et utilisée pendant toute la durée de vie du botnet. Ceci est le cas du botnet Zeus. Par conséquent, si cette clé est découverte, les chercheurs peuvent analyser les messages échangés et infiltrer le botnet. Ainsi, après la découverte de la clé de chiffrement du botnet Storm, des chercheurs ont réussi à l'infiltrer et à injecter des commandes dans le botnet [LWM09].

Le déchiffrement d'un canal chiffré nécessite un effort considérable pour chaque botnet, parce qu'il faut effectuer dans la majorité des cas de l'ingénierie inversée dans le binaire du logiciel malveillant. De plus, les créateurs de botnets mettent en oeuvre des mécanismes dans leur fichier binaire pour retarder l'opération d'ingénierie inversée. En conséquence, l'opération de déchiffrement d'un canal chiffré est onéreuse et ne peut être appliquée qu'à une minorité de botnets.

1.3 Attaques et impacts

L'objectif principal d'un opérateur de botnet est d'utiliser les machines infectées afin de déclencher des attaques. Ces attaques ciblent soit les données des utilisateurs des machines infectées, soit des cibles distantes en utilisant ces machines infectées.

Les opérateurs de botnets sont motivés par la génération de profits. Pour cela, ils utilisent toutes les ressources des machines infectées pour enclencher des attaques.

Nous allons détailler ci-dessous les principales attaques générées par les botnets :

1.3.1 DDoS

Le but d'une attaque par déni de service est de rendre le service visé inaccessible. Pour cela, l'attaquant envoie des paquets à la destination de la cible afin de perturber son fonctionnement et de la rendre incapable de répondre adéquatement aux utilisateurs légitimes. Dans une attaque par inondation, l'attaquant cherche à saturer le lien réseau de la cible afin de la rendre inaccessible aux utilisateurs légitimes.

Lorsque plusieurs machines participent à une attaque par déni de service vers la même cible, elle est appelée attaque déni de service distribué ou DDoS (Distributed Denial of Service). Différents outils d'attaques DDoS sont apparus à la fin des années 90 [LRST00] telle que Trinoo, TFN (Tribe Flood Network), Stacheldarht, ou Shaft.

Des services de location de botnets spécialisés dans les attaques DDoS sont apparus. Le prix de cette location dépend de différents paramètres : la durée de l'attaque, de sa sophistication, et de la taille de la cible. En 2011, d'après une annonce de ce genre de service, le nombre de machines permettant de mettre hors ligne un serveur moyen [Loe12] était de 300 $. Le cout de cette location était de 150 $ par jour.

Des kits de création de botnets spécialisés dans les attaques par déni de service ont été développés tels que BlackEnergy [Naz07] et Darkness [dar]. Ils offrent à leurs clients une interface web afin de leur permettre de coordonner les machines infectées et lancer les attaques DDoS. Ils peuvent générer différents types d'attaques tels que l'inondation UDP ou ICMP.

Lorsque l'opérateur d'un botnet décide de lancer une attaque par déni de service contre une cible, il envoie des commandes d'attaques vers les machines infectées. L'attaquant précise dans ces commandes le type et la durée de l'attaque qu'il veut lancer. Les machines infectées et membres de ce botnet, après la réception de ces commandes, envoient des paquets d'attaques vers la cible.

Dans certaines attaques par déni de service tel que la saturation du lien réseau, la machine d'attaque peut ne pas utiliser son adresse dans les paquets envoyés à la cible. En effet, elle peut usurper une adresse IP puisqu'elle n'attend pas de réponse particulière du serveur ciblé. La conséquence de ce type d'attaque est que le service attaqué ne pourra pas identifier de manière simple les machines à l'origine de cette attaque.

Les motivations des attaques par déni de service sont diverses [MR04] : politique, chantage, ou vengeance.

- Politique : Le but de ces attaques est de faire passer un message politique. On peut citer les attaques contre les infrastructures importantes de l'Estonie en 2007 [Les07], et la Georgie en 2008 [Naz08]. La protestation peut également prendre la forme d'un DDoS comme l'attaque du groupe Anonymous contre des services de paiement en ligne en 2010 [PSM+10].
- Chantage : Le but des attaquants est de se faire rémunérer auprès de la cible. Pour cela, soit ils lancent une attaque et demandent à la cible de payer afin de l'arrêter, soit les attaquants demandent à la cible de le payer pour ne pas lancer l'attaque. Les victimes sont en générale des entreprises dont l'activité commerciale dépend de la disponibilité de l'accès internet tel que les sites de pari en ligne.
- Vengeance : Le but de ces attaques est de mettre hors service un concurrent. En effet, le fondateur de la compagnie de payement en ligne Russe ChronoPay [GN12] a été arrêté par la police à cause de son implication dans une attaque par DDoS contre un concurrent.

Les attaques par déni de service représentent une menace non négligeable pour les opérateurs réseau et pour les entreprises connectées à internet. Cette menace devrait croitre dans les prochaines années au vu des débits atteints en 2013. Ainsi, le débit d'une attaque de déni de service contre Spamhaus [Pri13] a dépassé 300 Gps. Bien que ces débits soient importants, la politique de surdimensionnement de la bande passante mis en oeuvre par les opérateurs réseau leur permet toutefois de supporter encore ces attaques même si les cibles peuvent être inaccessibles.

1.3.2 Spams

Les courriers électroniques indésirables, ou spams, représentaient en 2012 80% des mails échangés [SSPS12]. 88% des spams sont envoyés par des botnets [SGHSV11]. Ainsi, le botnet Pushdo/Cutwail qui comprenait en moyennes 15000 machines, a envoyé 87.7 milliards de spams en 26 jours. Les chercheurs ont estimé que les profits générés par les opérateurs de ce botnet entre juin 2009 et aout 2010 étaient situés entre 1.7 et 4.2 millions de dollars américains. En outre, le botnet Kelihos a envoyé 3.8 milliards de spams par jour en 2011 [kel].

Dans ce type de botnet, l'opérateur définit un modèle [SGHSV11] de messages à

envoyer ainsi qu'une liste d'adresses mails de cibles potentielles. Les machines infectées génèrent les courriers indésirables à partir du modèle du message reçu et l'envoient à destination des adresses ciblées. La valeur ajoutée d'un botnet dans l'envoi de spam est la disponibilité des adresses IP des machines infectées qui permet de limiter l'efficacité des mécanismes de filtrage de spams.

Les opérateurs de botnets sont rémunérés pour l'envoi de messages indésirables, ces derniers contenants des annonces publicitaires de produits ou de services illicites comme les produits pharmaceutiques contrefaits. Les spams sont aussi des vecteurs d'escroqueries en tout genre et de tentatives de phishing.

Les spams peuvent également être comptabilisés comme des vecteurs d'infection. Les courriers indésirables peuvent contenir des programmes d'infection en pièce jointe, ou des liens vers des pages web d'exploits permettant d'infecter la machine des personnes cliquant sur ces liens.

Les botnets spécialisés dans l'envoi de spams sont loués à des tiers afin de leur permettre de diffuser leurs messages. Le prix de cette location dépend [SGHSV11] de la location géographique des machines infectées qui vont envoyer les messages indésirables et de la réputation de leurs adresses IP. En effet, une adresse IP non utilisée auparavant pour envoyer des spams est vendue plus cher qu'une autre adresse IP utilisée puisque sa réputation est moins bonne.

Pour contourner le filtrage des spams mis en oeuvre dans les serveurs mails, des opérateurs de botnet ont intégré des modules de filtre antispam dans les logiciels malveillants. Ainsi, le logiciel malveillant se base sur ce filtre pour adapter le contenu des courriers indésirables afin qu'il ne soit pas identifié par le filtre de la cible.

Même si la majorité des spams sont filtrés à la destination, leur acheminement représente une charge non négligeable pour les liens réseaux de l'opérateur. De plus, le filtrage de spams représente aussi une charge de travail supplémentaire pour les serveurs mails.

1.3.3 Vol de données

La multiplication des données sensibles stockées dans les machines des utilisateurs a attisé la convoitise des cybercriminels. Ainsi, des données comme les identifiants de messagerie, et les numéros de carte de crédit peuvent être revendues à des tiers.

Pour cela, les cybercriminels ont conçu des logiciels malveillants leur permettant d'avoir un contrôle total sur la machine de la victime. Ces logiciels peuvent récupérer des informations directement dans la mémoire ou le disque de la machine ou en interceptant les informations entrées au clavier ou à la souris par la victime.

Les logiciels malveillants de botnet tels que Zeus [BOB+10], Torpig [SGCC+09], et SpyEye [SEB12] opèrent directement dans le navigateur web de la victime. Ainsi, lorsque cette dernière remplit le formulaire d'une page web, le logiciel malveillant cherche à subtiliser des informations sensibles telles que le numéro de compte et le mot de passe de l'utilisateur. Ce type d'attaque est appelé Man-In-The-Browser. La particularité de cette attaque est qu'elle permet d'intercepter les informations échangées dans une session sécurisée avant son chiffrement par le navigateur web.

Les informations subtilisées aux victimes sont ensuite acheminées vers l'opérateur du botnet. Ce dernier les revend à d'autres cybercriminels. En 2008, d'après le rapport de Symantec sur l'économie clandestine des cybercriminels [FJT+08], un numéro de carte de crédit était revendu entre 0.5 et 12$, et un couple identifiant mot de passe de messagerie entre 4 et 30$.

La prise en main des serveurs de contrôle et commande pendant dix jours du botnet spécialisé dans le vol de donnée, Torpig en 2009 [SGCC+09] a permis à des chercheurs d'analyser les données de 180000 machines contrôlées par ce botnet. Ils ont pu ainsi collecter 70 GB de données contenant une multitude de numéros de carte de crédit, et d'identifiants bancaire. En plus, ils ont collecté 297962 paires d'identifiant et mots de passe pour différents sites web. La valeur de revente des informations bancaires collectées peut atteindre 8.3 millions de dollars d'après les chercheurs.

1.3.4 Fraude au clic

La publicité en ligne permet de financer les sites web qui proposent des services gratuits pour les visiteurs. Ces sites sont rémunérés suivant deux critères : le nombre de visiteurs d'une page web qui ont par la même occasion aperçu l'annonce, et le nombre de clics sur cette annonce.

Les revenus de la publicité en ligne ont augmenté de 17% entre 2012 et 2013 [IAB14]. Ils sont passés de 36.5 en 2012 à 42.8 milliards de dollars américains en 2013. Ainsi, ils sont devenus en 2013 le premier secteur de publicité devant la télévision publique.

Les opérateurs de botnets utilisent leurs machines infectées afin de générer des profits sur la publicité en ligne. Pour cela, l'opérateur du botnet demande à une entreprise de publicité d'afficher des annonces sur son site internet. Dés qu'il reçoit des liens vers des annonces publicitaires, il demande aux machines infectées de faire des requêtes sur ces liens ou de cliquer sur une bannière. En effet, l'utilisation des machines infectées permet de faire croire aux annonceurs que plusieurs utilisateurs légitimes ont visionné leurs annonces.

La fraude au clic représente une menace importante pour les annonceurs ainsi que pour leurs clients. En effet, d'après l'entreprise spécialisée dans la publicité en ligne "Click Forensics", la fraude représentait au dernier trimestre de 2010 19.1% de la totalité des clics [CdQC12].

1.3.5 PPI

Certains opérateurs de botnets se sont spécialisés dans la revente de machines infectées à d'autres opérateurs de botnets. Ces derniers achètent ainsi un accès à ces machines infectées afin d'y installer le logiciel malveillant de leur propre botnet.

Ce service de revente de machines infectées est appelé PPI (Pay Per Install). Ces botnets ont pour seule fonction d'exécuter des programmes. Ainsi, lorsqu'une machine est vendue, l'opérateur lui demande d'exécuter le logiciel malveillant du botnet de l'acheteur. Les botnets de ce type les plus connus sont LoaderAdv, Goldinstall, Virut, et Zlob [CGKP11].

Une machine infectée peut se retrouver membre de plusieurs botnets simultanément [CGKP11] puisque le botnet du logiciel malveillant installé peut lui même être un botnet de type PPI. Ainsi, dans une étude des botnets de type PPI, une machine s'est retrouvée infectée par 6 botnets différents. En conséquence, la machine infectée se connecte à une multitude de serveurs de contrôle opérés par différents attaquants.

Le prix des machines infectées varie selon leur position géographique. Plus le niveau de vie du pays de la victime est élevé, plus le prix de vente de la machine est élevé. En effet, le prix de 1000 machines infectées varie entre 100 et 180$ pour l'Europe et les États-Unis, et entre 7 et 8$ pour l'Asie.

1.3.6 Fast-flux

La technique Fast-flux est utilisée par l'attaquant pour dissimuler l'adresse IP d'un serveur malveillant. Le nom de domaine malveillant est associé à une liste d'adresses IP dynamique. Ces dernières jouent le rôle de proxy inverse ou d'intermédiaires vers le serveur malveillant. En conséquence, l'adresse IP du serveur est cachée.

Ces proxys inverses sont des machines infectées contrôlées par un opérateur de botnet. Ils sont utilisés pour cacher des serveurs de fishing, d'escroquerie, ou des serveurs de distribution de logiciels malveillants. Ces botnets peuvent gérer simultanément plusieurs noms de domaine malveillants [NH08].

Des botnets non spécialisés dans le fast-flux peuvent utiliser une partie de leurs machines infectées afin de jouer le rôle de reverse-proxy. Ainsi, le botnet Storm utilisait 1% de ses machines infectées afin d'assurer le service fast-flux de ses serveurs de contrôle. Le nombre de machines infectées associées à un nom de domaine du botnet Storm en 2008 [NH08] était de 100000 pour une période de 60 jours.

1.3.7 Escroquerie

Les logiciels malveillants ont un contrôle total sur la machine de la victime et l'ensemble de ses ressources. De plus, ils peuvent altérer le bon fonctionnement de cette machine. Dans les botnets de ce type, les logiciels malveillants rançonnent la victime pour rétablir le bon fonctionnement de la machine.

Nous pouvons citer quelques exemples de botnets d'escroqueries :

- Les faux antivirus : Le logiciel malveillant exécute un programme dans le système de la victime qui se présente comme étant un antivirus. Il demande à la victime de payer pour accéder à la version complète de cet antivirus ou pour nettoyer son système.
  Les revenus de 3 botnets de ce type ont été estimés par des chercheurs [SSGA+11] à 130 millions de dollars américains en 2010.
- Les ransomeware : Le logiciel malveillant bloque le démarrage de la machine de la victime. En se faisant passer par une autorité officielle telle que la gendarmerie ou la police, il lui affiche un message indiquant que la victime a commis un délit répréhensible par la loi, et lui demande de payer une amende en utilisant sa carte de crédit. Après le paiement de l'amende, la machine de la victime démarre normalement. De plus, certains logiciels malveillants de ce type chiffrent le disque dur de la victime [Ler13] pour l'obliger à payer la rançon.

1.3.8 Scans

Des botnets tels que Conficker [DKP+12] ajoutent des fonctionnalités de cartographie du réseau dans leur logiciel malveillant afin de trouver d'autres machines vulnérables. Les vulnérabilités de ces machines sont alors exploitées afin de les faire rejoindre le botnet.

Pour cela, les machines infectées envoient des paquets vers des ports réseau spécifiques afin de déterminer si une application vulnérable est en cours d'exécution sur la machine distante. En analysant la réponse ou (l'absence de la réponse), le membre du botnet détermine si l'application visée est en cours d'exécution.

Le scan réseau peut être classé en deux catégories [ZLK10], vertical et horizontal :

- scan vertical : le but de ce scan est d'exploiter une application vulnérable prédéfinie. En conséquence, un seul numéro de port est examiné dans les machines du réseau afin de trouver celles qui exécutent l'application vulnérable.

– scan horizontal : l'attaquant possède plusieurs exploits qui lui permettent d'exploiter les vulnérabilités de plusieurs applications. Ainsi contrairement au scan vertical, il examine plusieurs ports sur un ensemble de machines.

1.3.9 Autres attaques

Les botnets exploitent les ressources des victimes pour une multitude d'attaques.
– Les ressources de calcul des machines infectées peuvent être exploitées afin de générer de la monnaie virtuelle comme des bitcoins [PGP12].
– Les adresses IP des machines infectées sont exploitées pour cacher l'emplacement d'attaquant. En effet, les attaquants utilisent plusieurs proxys ou tremplins afin de cacher son adresse IP à la cible [IH05].

1.4 Conclusion et perspective

Dans ce chapitre, nous avons présenté les différentes menaces et problématiques posées par les botnets. Nous avons également analysé les différentes méthodes de communication au sein d'un botnet et leurs implications par rapport aux attaques générées. Ainsi, nous avons classifié les botnets par rapport à ces méthodes de communications : botnet HTTP, et botnet IRC. En revanche, nous avons remarqué que, quels que soient les protocoles ou les méthodes de communication utilisées, les botnets se basent en grande majorité sur le service DNS pour atteindre leur serveur de contrôle.

Les opérateurs de botnet utilisent les machines infectées sous leur contrôle pour enclencher une multitude d'attaques. Certaines de ces attaques ne sont pas détectables au niveau réseau, puisqu'elles ont lieu au niveau de la machine, comme le vol de données.

Pour lutter contre les attaques menées par les botnets au niveau réseau, il est nécessaire d'identifier dans un premier temps les sources de ces attaques afin d'appliquer des méthodes de remédiation. Les sources de ces attaques correspondent à des machines infectées contrôlées par un botnet par l'intermédiaire d'un serveur de C&C. La remédiation de ce serveur empêche les commandes d'attaques d'atteindre les machines infectées.

Chapitre 2

Traçabilité du trafic internet

Une des principales attaques réseau menées par les botnets est le déni de service distribué (DDoS). Durant cette attaque, plusieurs machines envoient des paquets vers un service afin de le rendre inaccessible aux utilisateurs légitimes.

L'utilisation de plusieurs machines dans l'attaque DDoS permet d'amplifier le résultat de l'attaque. Le débit de l'attaque générée auprès de la cible correspond à la somme des débits des machines participantes. Il est difficile de filtrer le trafic de chaque machine d'attaque, puisque ce trafic peut être considéré comme légitime par leur réseau.

Pour remédier à ces attaques, il est nécessaire d'identifier leurs sources, c'est-à-dire les machines infectées membres du botnet qui a déclenché l'attaque. Une identification basée sur l'adresse source des paquets d'attaques n'est pas suffisante à cause de l'usurpation d'adresses IP. Les machines d'attaques, dans certains réseaux, peuvent usurper leur adresse pour ne pas se faire identifier ou pour amplifier le débit de l'attaque. Elles n'utilisent pas leur adresse IP réel dans les paquets d'attaques.

La traçabilité IP répond à la problématique de l'identification de la source de l'attaque. Elle est définie [BA03a] comme étant le processus d'identification de la source de n'importe quel flux réseau sans tenir compte des adresses IP utilisées. Elle fait partie des moyens de lutte contre les attaques par déni de service qui menacent la disponibilité du réseau.

L'application d'une méthode de traçabilité IP ne stoppe pas une attaque DDoS en cours. Elle permet à la cible ou aux opérateurs réseau de remonter au plus près des attaquants pour atténuer le débit de l'attaque. Par conséquent, le lien réseau de la cible sera moins saturé.

Les attaques par déni de services ont aussi des conséquences sur les machines infectées qui y participent et sur l'opérateur réseau de la cible. Le lien réseau de la machine infectée peut être ralenti ou complètement saturé puisqu'il est utilisé pour envoyer les paquets d'attaques. De plus, les liens réseau de l'opérateur de la cible peuvent aussi être saturés. Par conséquent, le voisinage de la cible peut aussi endurer l'attaque.

Ces travaux de traçabilité du trafic internet sont la continuation de ceux entamés durant mon stage de master sous la direction du professeur Ahmed Serhrouchni [Hac10].

2.1 Analyse des attaques par déni de service

Une attaque par déni de service distribuée est constituée de différents acteurs comme illustrés dans la figure 2.1 : l'attaquant, le serveur de contrôle et commande (C&C) du botnet, les machines infectées, et la cible de l'attaque. L'attaque par déni de service

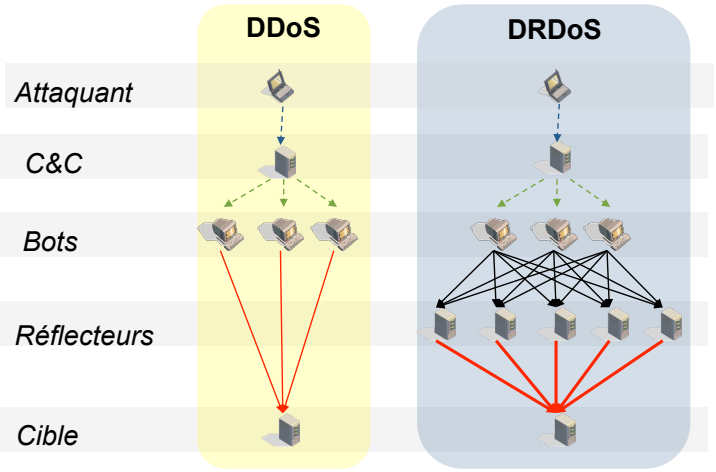

FIGURE 2.1 – Attaque par déni de service directe et réflective

indirecte distribuée inclut des machines réflectives :

- L'attaquant : enclenche l'attaque. Il décide de sa durée, de sa nature (le type de l'attaque, le nombre de machines infectées qui y participent, son débit, l'adresse de la cible, le service ou le port réseau visé). Il transmet les commandes d'attaque aux machines infectées choisies pour participer à l'attaque au travers du serveur de contrôle et commande (C&C).
  Afin de cacher son adresse, l'attaquant peut utiliser des machines tremplins "stepping stones" lors de sa connexion avec le serveur de C&C. Il se connecte à chaque machine tremplin avec l'adresse IP de la machine précédente. Ainsi, il se connecte au serveur de contrôle et commande avec l'adresse de la dernière machine tremplin.
- Le serveur de contrôle et commande (C&C) : coordonne la communication entre les machines compromises ou bots avec l'attaquant. Il renvoie les commandes d'attaques aux membres du botnet. Il collecte ensuite les rapports des différentes attaques auprès des machines infectées. En fin de compte, il agrège ces rapports afin de les présenter à l'opérateur du botnet.
- Bots : envoient des paquets d'attaques vers la cible afin de rendre le service visé inaccessible aux utilisateurs légitimes. Il entreprend l'attaque à la suite de la réception des commandes spécifiques de la part du serveur de C&C. Elles renvoient au serveur de C&C les rapports sur le déroulement de l'attaque afin d'informer l'opérateur du botnet.
- Cible : reçoit les paquets d'attaques et collecte les informations sur l'attaque à des fins d'analyse.
- Machines réflectives : sont utilisées dans les attaques par déni de service réflectives (DRDoS). Ces machines sont accessibles depuis l'extérieur, c'est-à-dire qu'elles ont une adresse IP publique. Leur rôle est de cacher l'adresse IP des machines d'attaques et membres du botnet, ou pour amplifier le débit de l'attaque.

Plusieurs chercheurs ont proposé ces dernières années des taxonomies des attaques par déni de service [MR04, FS05]. Dans ce chapitre, nous proposons de classer les attaques par déni de service par rapport à leur direction, voire figure 2.1 : attaque directe,

attaque indirecte ou réflective.

- Attaque directe : Dans ce type d'attaque, les machines corrompues envoient des paquets directement vers la cible. Pour cela, elles utilisent dans le champ "adresse destination" de l'entête des paquets d'attaques l'adresse de la cible. Par ailleurs, pour certaines attaques, elles peuvent usurper leur adresse, c'est-à-dire qu'elles n'utilisent pas leurs adresses IP dans le champ "adresse source" de ces paquets. On peut citer par exemple les attaques par inondation qui visent à saturer le lien réseau de la cible où les machines d'attaque peuvent usurper leur adresse IP.
  Les attaques par déni de service directes peuvent viser des services de la couche application. On peut citer l'attaque Slowloris [RL] contre les serveurs web de type Apache. Le but de l'attaque est d'ouvrir le maximum de connexions avec le serveur web en envoyant régulièrement des paquets vides afin de maintenir la connexion ouverte. Ceci empêche le serveur web d'ouvrir de nouvelles connexions. Par conséquent, les utilisateurs légitimes ne peuvent pas se connecter au serveur.
- Attaque indirecte ou réflective : Dans ce type d'attaque, les paquets à destination de la cible transitent par des machines tierces [Pax01]. Cela afin d'amplifier le débit de l'attaque ou pour cacher l'adresse de l'attaquant. Ces machines tierces ne sont pas corrompues ou membres d'un botnet. Elles ignorent leur participation dans des attaques par déni de service. Elles sont appelées machines réflectives. Les attaques réflectives les plus connues sont les attaques SMURF et les attaques DNS réflectives.
  Dans l'attaque SMURF [Kum07] [CA98], les machines d'attaque envoient des requêtes ICMP [Pos81] vers des machines tierces en utilisant dans le champ "adresse source" de l'entête du paquet l'adresse IP de la victime. À la suite de la réception de ces requêtes, les machines réflectives envoient des réponses à la cible de l'attaque en croyant répondre au demandeur.
  L'attaquant peut amplifier le débit de son attaque en envoyant ses paquets vers l'adresse broadcast du réseau puisque le filtrage de ce type de paquet n'est pas appliqué dans certains réseaux mal-configurés. Lorsqu'un attaquant envoie une requête vers l'adresse broadcast d'un réseau réflecteur de 20 machines, la victime recevra 20 réponses de mêmes tailles que le paquet initial. Ainsi, l'attaquant a amplifié son attaque avec un facteur de 20.
  Dans les attaques DNS réflectives, les attaquants envoient des requêtes vers des serveurs DNS ouverts. Ces derniers acceptent de répondre aux requêtes des machines qui ne sont pas dans leur réseau. L'attaquant essaye d'amplifier son attaque en maximisant le taux de la taille de réponse par rapport à la taille de la requête. Pour cela, il envoie les requêtes DNS qui génèrent le plus d'informations dans la réponse. La plus utilisée est la requête de type "ANY" [UC13]. La réponse de cette requête renvoie toutes les informations sur une zone DNS.

Dans les attaques DDoS par inondation, il est difficile de se défendre ou de mettre en oeuvre des moyens d'atténuation au niveau de la cible parce que cette attaque vise à saturer son lien réseau. Si le débit de l'attaque dépasse la bande passante du lien réseau de la cible, cette dernière devient inaccessible aux utilisateurs légitimes qui essayeraient de se connecter dessus. Le débit de certaines attaques peut saturer le lien réseau de grandes organisations ou entreprises. Ainsi, une attaque DNS réflective en mars 2013 [Pri13] contre une organisation de lutte contre le spam a atteint un débit de 300 Gb/s.

La politique d'augmentation de la bande passante dans les réseaux ne permet pas de supporter des attaques qui atteignent ces débits. Il faudrait les atténuer au plus près de l'attaquant avant la convergence des différents flux distribués dans le cas d'attaques

directes, et avant leur amplification dans le cas d'attaques réflectives.

Une atténuation en amont d'une attaque DDoS permet aussi de protéger le réseau de l'opérateur de la cible. Le filtrage des paquets d'attaques est effectué soit chez l'opérateur de la cible ou chez les opérateurs réseau voisins. Ce filtrage permet à la cible d'être accessible aux membres du même réseau et de ceux des opérateurs réseau voisins. Si ces derniers collaborent dans le processus de filtrage des paquets d'attaques.

Afin de remédier aux attaques DDoS en cours, il faut identifier les sources de celles-ci. L'identification de ces sources est difficile si les adresses sources des paquets d'attaques sont usurpées. Cette usurpation est possible seulement si l'opérateur réseau n'applique pas la norme [FS] qui recommande au routeur du réseau de vérifier l'adresse IP source des paquets acheminés et de bloquer ceux ayant des adresses usurpées.

Des organisations [Spa] calculent la réputation des adresses IP en se basant sur celle qui participe à des attaques réseau telles que l'envoi des spams, ou le déni de service. Ces informations sont utilisées par les fournisseurs de services afin de prévenir certaines attaques en bloquant les adresses IP qui ont une mauvaise réputation. Par exemple, les serveurs de messagerie utilisent cette réputation pour identifier les courriers indésirables.

L'inclusion dans des listes noires peut motiver un botnet pour ne pas exposer directement ses membres et d'usurper leurs adresses IP lors d'attaques DDoS. En effet, les machines infectées sont revendues entre les opérateurs de botnets et une machine ayant une adresse IP dans une liste noire est revendue plus cher qu'une machine qui a une adresse dans cette liste [SGHSV11].

2.2 Analyse et limites des méthodes de traçabilités

La traçabilité est appliquée par les opérateurs réseau à travers les routeurs par où transitent les paquets. Elle peut ne pas être mise en oeuvre par l'ensemble des routeurs du réseau et permettre à la cible de remonter à la source de l'attaque. Par ailleurs, l'opérateur réseau peut utiliser des équipements supplémentaires pour mettre en oeuvre la traçabilité.

Les différentes méthodes traçabilité sont appliquées sur l'ensemble des flux du réseau. En effet, l'opérateur du réseau par où transitent les paquets ne peut pas deviner lors du routage si c'est un flux d'attaque ou un flux légitime.

La traçabilité est considérée comme un moyen de dissuasion d'attaques futures [Alj03]. Si l'attaquant a conscience qu'on peut remonter jusqu'à lui après une attaque, il se peut qu'il ne la déclenche pas.

Les approches de traçabilité peuvent être classées en 3 catégories : le log de paquets, le marquage de paquets, et la signalisation.

2.2.1 Logs de paquets

Dans cette approche, les routeurs enregistrent dans leur mémoire une partie des paquets qui transitent. Comme ils ne peuvent pas différencier les flux d'attaque des flux légitimes, ils enregistrent des informations sur l'ensemble des flux qu'ils acheminent.

Cette approche nécessite une coopération entre les différents routeurs du réseau pour pouvoir remonter à la source d'un flux. Les routeurs s'interrogent mutuellement afin de savoir si un flux d'attaque a transité par l'un d'entre eux. Ainsi, en plus du routage de paquets et le stockage des informations sur les flux, ils doivent communiquer entre eux pour savoir par où sont passés les paquets d'attaques.

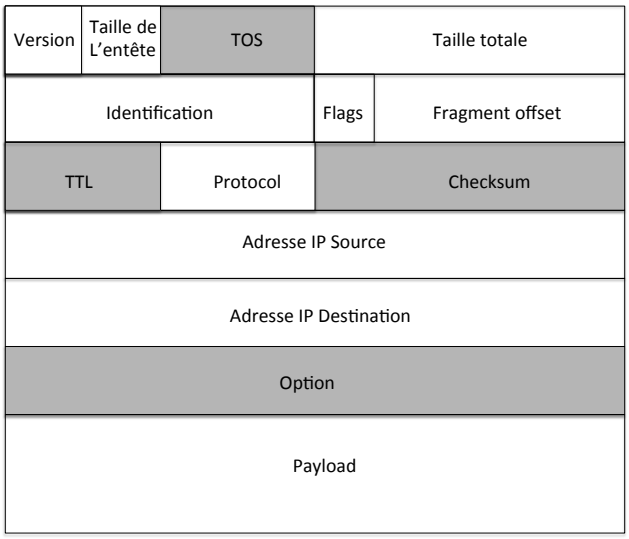

FIGURE 2.2 – L'entête d'un paquet IP. Les champs colorés en gris ne sont pas inclus dans le calcul du digest

Snoeren et al. ont proposé l'approche de traçabilité SPIE (Source Path Isolation Engine) [SPS+01]. Dans cette approche, les routeurs stockent des informations sur tous les paquets qu'ils acheminent. Pour pouvoir être appliquée à des routeurs d'opérateur réseau de large échelle et pour optimiser l'espace mémoire, ils utilisent des structures de données probabilistes, des filtres de Bloom [Blo70] pour représenter les paquets acheminés dans leur mémoire.

L'avantage principal de cette approche est de pouvoir remonter à la source d'un flux en utilisant un seul paquet. Ainsi lors d'une attaque par déni de service utilisant une adresse IP usurpée, un seul paquet d'attaques est suffisant pour remonter à la source de l'attaque. Cette approche permet de remonter au routeur le plus proche de l'attaquant ayant mis en oeuvre l'approche de marquage.

Les routeurs du réseau n'enregistrent pas les paquets entièrement dans leur mémoire. Un enregistrement de l'ensemble des informations contenues dans les paquets ne permet pas de tenir la charge du réseau. Ils enregistrent seulement un digest qui permet de représenter le paquet. Les routeurs calculent ce digest à partir des champs invariables de l'entête IP en plus du premier octet de la charge utile du protocole IP (Payload). Ces champs invariables ne sont pas modifiés par les routeurs lors de l'acheminement du paquet. Elles permettent de discerner le paquet, quelque soit sa position dans le chemin entre l'émetteur et le destinataire.

Les champs variables de l'entête IP lors du routage du paquet ne sont pas pris en compte dans le calcul du digest. Ils sont marqués en gris dans la figure 2.2. Les champs variables sont : le Type de service, le TTL, le checksum, et les options.

Lors de la détection d'une attaque, la cible, munie d'un paquet d'attaques ou de son digest, demande aux routeurs de son réseau qui participent au processus de traçabilité de vérifier s'ils ont acheminé ce paquet. Le routeur qui a acheminé le paquet demande à ses voisins s'ils ont fait de même. Cette opération est répétée jusqu'au routeur le plus proche de l'attaquant.

Des améliorations ont été apportées à la méthode proposée par Snoeren [SPS+01].

Li et al. [LSXL04] proposent une méthode de traçabilité basée sur les flux, contrairement à l'approche initiale basée sur les paquets. Dans cette méthode, les routeurs enregistrent les informations du premier paquet d'un flux seulement. Ceci permet d'améliorer l'utilisation de la mémoire des routeurs qui devaient enregistrer tous les paquets du flux. Ils doivent mémoriser les flux en cours d'acheminement afin de les stocker une seule fois.

La méthode SPIE citée précédemment est centralisée. Cette approche ne peut pas être appliquée dans un environnement distribué et multi opérateur tel qu'internet. Kormaz et al. [KGSD07] proposent une architecture de traçabilité entre systèmes autonomes (AS) de la méthode SPIE.

L'avantage principal de l'approche de log de paquets est qu'elle permet de remonter à la source d'un flux en utilisant un seul paquet. Dans les attaques distribuées, plusieurs machines envoient des paquets d'attaques vers la cible. Dans ces attaques, même si le nombre de paquets envoyés par chaque machine est faible, l'incidence de l'attaque peut être importante. Une approche de traçabilité qui se base sur un nombre faible de paquets pour reconstruire le chemin inverse de l'attaque est souhaitable pour remonter aux nombreuses sources. Le chemin inverse d'une attaque correspond à l'ensemble ordonné des routeurs entre la cible et l'attaquant.

Cette approche de traçabilité nécessite aux routeurs une mémoire de stockage importante, afin de sauvegarder tous les digests des paquets ou flux qu'ils acheminent. Dans la méthode SPIE [SPS+01] qui se base sur des filtres de Bloom pour représenter les digests des paquets, chaque routeur a besoin d'une mémoire de stockage d'au moins 0.5% de la capacité de son lien réseau.

Les informations de traçabilité sur les paquets acheminés ne peuvent pas être stockées indéfiniment par les routeurs. Les cibles d'une attaque doivent envoyer des requêtes de reconstruction du chemin d'attaque au moment de celle-ci afin de pouvoir la retrouver dans la mémoire des routeurs.

Les filtres de Bloom binaires utilisés dans l'approche de traçabilité permettent d'optimiser l'espace mémoire de stockage, mais elles ne permettent pas de supprimer des éléments représentés. Par conséquent, lorsque le filtre de Bloom est rempli, il est réinitialisé afin de pouvoir y ajouter de nouveaux éléments.

2.2.2 Marquage de paquets

Dans cette approche, les routeurs insèrent des informations de traçabilité dans les paquets acheminés. Les informations insérées permettent d'identifier les routeurs qui les ont acheminés. Ces informations sont insérées sur tout les paquets acheminés puisqu'ils ne peuvent pas faire la différence entre un paquet légitime et un paquet d'attaques.

Contrairement à l'approche de log de paquets où c'est aux routeurs du réseau de reconstruire le chemin d'un flux d'attaque, dans le marquage de paquet c'est à la destination de reconstruire ce chemin, en utilisant les informations insérées par les routeurs.

Dans une attaque par déni de service, la cible reçoit les informations de traçabilité des routeurs qui la séparent de l'attaquant et qui marquent les paquets. Elle utilise ces informations afin de construire le chemin inverse de l'attaque. La cible construit le chemin inverse complet de l'attaque, pour inférer quel routeur est le plus proche de l'attaquant.

Savage et al. [SWKA00] ont proposé une méthode de traçabilité basée sur le marquage qu'ils ont intitulé Fragment Marking Scheme (FMS). Dans cette méthode probabiliste, les routeurs insèrent les informations de traçabilité dans les entêtes des paquets avec une certaine probabilité définie par le système. Dans cette méthode, pour que tous les routeurs du chemin puissent marquer leurs informations de traçabilité lors d'une at-

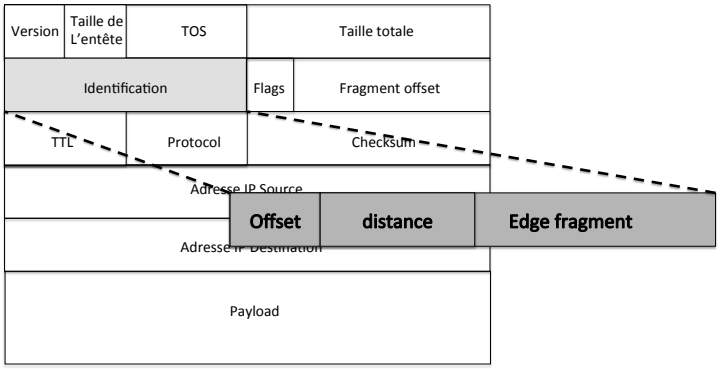

FIGURE 2.3 – Marquage des informations de traçabilité dans le champ identifier de l'entête IP

taque par déni de service, ils définissent une faible probabilité de marquage d'une valeur de 0.04.

La contrainte principale de l'approche de marquage est de ne pas rajouter de surplus de bande passante. Ceci implique que les routeurs réutilisent des champs de l'entête IP afin d'insérer leurs informations de traçabilité. La majorité des méthodes de marquage utilisent les champs de l'entête IP non fréquemment utilisés pour insérer leurs informations de traçabilité.

Par exemple, la méthode FMS utilise le champ "identifiant de fragment" de l'entête IP, voir la figure 2.3. Lorsqu'un message est fragmenté, le destinataire utilise le champ "identifiant de fragment" pour joindre les fragments du même message. Par contre, si le message n'est pas fragmenté, ce champ n'est pas examiné par les routeurs qui acheminent le paquet et le destinataire.

La taille du champ "identifiant de fragment" est de seulement 16 bits. Ce qui est inférieur à la taille d'une adresse IP qui est de 32 bits. Ainsi, afin que la cible de l'attaque puisse recevoir la totalité des adresses IP des routeurs, ces derniers marquent de manière aléatoire une partie de leur adresse IP. De plus, les routeurs insèrent d'autres informations afin de permettre au destinataire de reconnaitre la position du routeur dans le chemin de l'attaque.

Ainsi, un routeur a besoin de plusieurs paquets pour envoyer la totalité de son adresse IP. Puisque le chemin entre l'émetteur et le destinateur d'un flux contient plusieurs routeurs, il faut un nombre important de paquets pour recevoir la totalité des informations insérées. Ceci correspond dans le cas d'une attaque au nombre de paquets nécessaires pour reconstruire le chemin de l'attaque. En effet, dans la méthode FMS, la cible d'une attaque par déni de service a besoin d'un millier de paquets [SWKA00] pour reconstruire le chemin de l'attaque.

Dans l'approche FMS et comme illustré dans la figure 2.3, les informations de marquage sont constituées de l'edge fragment, l'offset, et de la distance :

- Edge Fragement : une partie de l'adresse IP du routeur couplé au moyen d'un OU exclusif à la même partie de l'adresse IP du routeur suivant afin de lutter contre l'usurpation de marquage,
- l'offset de cette partie, c'est-à-dire la partie insérée de l'adresse IP du routeur

– et la distance qui sépare le premier routeur du destinataire.

Cette distance est initialisée à 0 lorsque le paquet est marqué, c'est-à-dire lors l'insertion d'information de traçabilité dans l'entête. Les routeurs suivants, incrémentent cette distance de 1 lors de l'acheminement du paquet. Par conséquent, la valeur de la distance correspond au nombre de routeurs entre le destinataire et le routeur ayant effectué le marquage.

Plusieurs méthodes probabilistes de marquage ont été proposées pour améliorer l'approche FMS. Dean et al. [DFS02] ont proposé une méthode algébrique afin d'encoder les adresses IP des routeurs dans un polynôme. Ils ont proposé d'étendre le marquage à d'autres champs de l'entête IP tel que le TOS (Type de service).

Des approches de traçabilité basées sur le marquage du hash de l'adresse IP telle que Advanced & Authenticated Marking Schemes (A&AMS) [SP01] et Fast Internet Traceback (FIT) [YPS05]. Ces méthodes supposent que la cible ou le destinataire d'un flux ont la capacité de connaitre les adresses IP et l'emplacement des routeurs en amont. Puisque la taille d'un hash est inférieure par rapport à la taille d'une adresse IP, il faut à la destination moins de paquets pour reconstruire le chemin de l'attaque par rapport à la méthode FMS.

En plus du marquage du hash de l'adresse IP, A&AMS permet l'authentification des marquages effectués par les routeurs. Ceci devrait augmenter le surplus de calcul nécessaire pour les routeurs au moment du marquage et aux destinataires des flux pour rechercher les clés de chiffrement utilisées.

Belenky et al. [BA03b] ont introduit une méthode de marquage déterministe. Contrairement à l'approche probabiliste où tous les routeurs du réseau participent au marquage, dans l'approche déterministe, il est effectué uniquement par les routeurs de bord de réseau. Ils insèrent leurs informations dans l'entête IP de manière déterministe. Par conséquent, le destinataire peut identifier le point d'entrée du flux dans le réseau.

En plus du champ "identifiant de fragment" pour marquer l'adresse IP des routeurs, l'approche déterministe utilise le premier bit du champ flags qui est inutilisés actuellement. Ce bit permet de différencier entre les parties de l'adresse IP marquée puisqu'elle est transmise en deux morceaux.

Dans l'approche déterministe, le destinataire a besoin d'un minimum de 2 paquets pour reconstruire l'adresse IP du routeur de bord du réseau qui a effectué le marquage. Ils ont démontré dans leur article [BA03b] que le destinataire peut reconstruire l'adresse IP avec une probabilité de 99% à la réception de 7 paquets.

L'avantage des approches de marquage de paquets est qu'elles n'engendrent pas de surplus de bande passante. La taille des paquets n'augmente pas après le marquage puisque les routeurs utilisent des champs inutilisés de l'entête du paquet IP pour insérer leurs informations.

Dans les approches de marquage de paquets, c'est au destinataire d'un flux donné de reconstruire le chemin inverse. Pour cela, il analyse les informations incluses par les routeurs traversées par les paquets du flux. Ceci permet de soulager les routeurs de coeur du réseau de la fonction de reconstruction pour qu'ils s'occupent uniquement de l'insertion d'information de traçabilité dans l'entête du paquet.

L'approche probabiliste citée précédemment permet une implémentation incrémentale dans le réseau. Cela signifie qu'elle est valide même si elle n'est pas appliquée à l'ensemble des routeurs du réseau ; Cela dit, plus le nombre de routeurs utilisant l'approche de marquage augmente, plus le chemin inverse du flux calculé par le destinataire sera précis.

Un inconvénient des approches du marquage, qu'elles soit probabiliste ou détermi-

niste, est qu'elles utilisent le champ "identifiant de fragment" de l'entête IP afin d'insérer des informations de traçabilité. Ainsi, puisque ce champ est utilisé par les destinataires pour reconstruire les paquets fragmentés, la fragmentation ne pourra pas être utilisée dans le réseau. En conséquence, si un paquet est fragmenté, la victime ne pourra pas le reconstruire ces différentes parties, puisque la valeur du champ "identifiant de fragments" sera différente pour ces parties.

2.2.3 La signalisation

Dans cette approche, les routeurs, pour envoyer des informations de traçabilité au destinataire d'un flux, génèrent un nouveau paquet. Le destinataire peut reconnaitre le routeur qui lui a envoyé ces informations. Contrairement à l'approche de marquage de paquet, la signalisation génère un surplus de bande passante.

L'approche de signalisation a été proposée par Bellovin et al. [BLT03] dans iTrace. Les routeurs génèrent des paquets ICMP appelés "ICMP Traceback Message" pour accompagner un flux réseau à son destinataire. Ces paquets contiennent des informations de traçabilité, qui sont l'adresse IP de l'interface d'entrée du paquet, et l'horodatage de la construction du paquet ICMP.

Comme dans l'approche de marquage de paquets, dans la signalisation, c'est au destinataire de reconstruire le chemin inverse du flux. En recevant suffisamment de messages de traçabilité de la part des routeurs du chemin, le destinataire peut identifier l'adresse ainsi que la position des routeurs qui le sépare de l'émetteur du flux.

La cible d'une attaque par déni de service utilise les informations de traçabilité reçue pour reconstruire le chemin de l'attaque. Ce chemin correspond aux routeurs entre le destinataire et l'émetteur. Ceci lui permet de remonter au routeur le plus proche de l'attaquant et qui a envoyé des paquets de signalisation.

Afin de limiter le surplus de bande passante engendré par le processus de traçabilité, chaque routeur du chemin qui a mis en oeuvre l'approche iTrace envoie un paquet de signalisation ICMP tous les 20000 paquets vers le destinataire du flux en cours. Cette faible probabilité permet aux routeurs de limiter le surplus réseau.

Des améliorations ont été apportées à la méthode iTrace, Mankin et al. [MMW+01] ont proposé une amélioration de la décision de génération des messages ICMP de traçabilité pour minimiser le nombre de paquets nécessaires à la reconstruction du chemin inverse d'un flux. La méthode ICMP Caddie (iCaddie) [WS04] permet de réduire le surplus de bande passante généré en envoyant moins de messages de signalisation au destinataire. En plus, elle permet aux routeurs de chiffrer leur marquage afin de diminuer l'usurpation de marquage par un attaquant.

La méthode iCaddie a besoin de moins de paquets de signalisation par rapport à la méthode iTrace parce qu'elle permet à plusieurs routeurs d'insérer leurs informations de traçabilité dans le même message de signalisation.

Ainsi, lorsqu'un message de signalisation est envoyé vers le destinataire, les routeurs du chemin insèrent aussi leurs informations de traçabilité s'ils le souhaitent.

L'inconvénient principal de l'approche de signalisation est qu'elle génère un surplus de bande passante alors que lors d'une attaque par déni de service le lien réseau de la cible est saturé.

L'autre inconvénient de l'approche de signalisation est que le destinataire d'un flux reçoit plus de messages de signalisation des routeurs les plus proches de lui par rapport aux routeurs proches de l'émetteur [BA03a]. En revanche, dans une attaque par déni de service, la cible a besoin de recevoir les informations de traçabilité des routeurs les plus

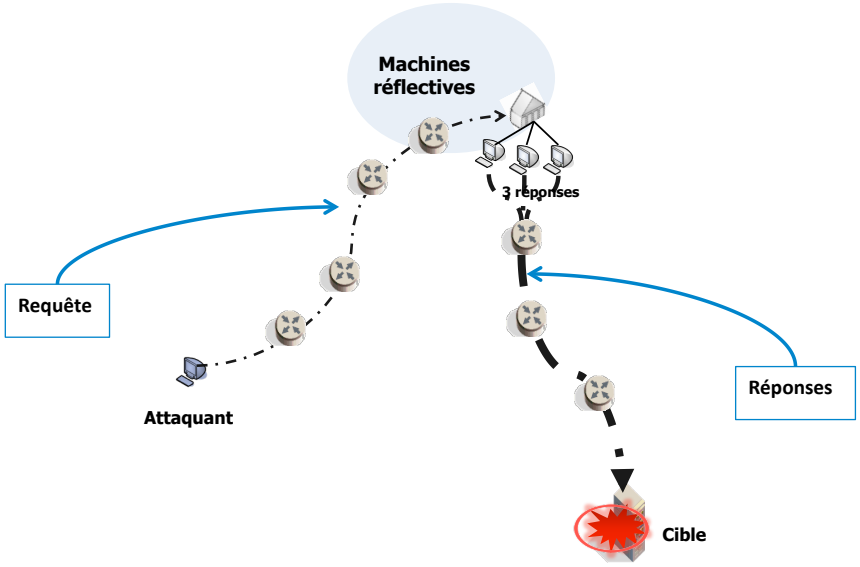

FIGURE 2.4 – Chemin d'une attaque SMURF

proches de l'attaquant pour localiser ce dernier.

2.2.4 Traçabilité des attaques réflectives

Les approches de traçabilité classiques décrites ci-dessus ne permettent pas de reconstruire le chemin d'attaques par déni de service réflectives. Elles permettent seulement de remonter à la source d'attaques directes, ou dans le cas d'une attaque réflective de remonter aux réflecteurs qui correspondent à des machines non contrôlées par l'attaquant et qui sont utilisés à leur insu pour envoyer des paquets en direction de la cible.

Nous avons basé notre analyse des attaques réflectives sur l'attaque ICMP réflective «SMURF». Cette attaque est basée sur le couple requête/réponse ICMP "echo request" et "echo reply", ainsi que sur des machines réflectives comme illustré dans la figure 2.4.

SMURF est une attaque par déni de service réflective qui vise à saturer le lien réseau de la cible avec des paquets ICMP. La machine d'attaque envoie des requêtes ICMP vers une machine réflective en utilisant comme adresse source celle de la cible. En réaction, la machine réflective envoie une réponse à la cible en croyant répondre à l'émetteur de la requête. Ceci permet à l'attaquant de masquer sa localisation par rapport à la cible.

Dans l'attaque SMURF, la taille de la réponse envoyée à la cible est égale à celle de la requête envoyée par l'attaquant. La taille des paquets ICMP "echo request" ou "echo reply" est limitée par le protocole IP. Elle peut atteindre 65507 octets.

L'impact de l'attaque SMURF peut être amplifié auprès de la cible. Au lieu d'envoyer la requête à une seule machine réflective, l'attaquant l'envoie vers l'adresse broadcast d'un réseau. La requête est ainsi reçue par un ensemble de machines. Bien que l'attaquant ait envoyé une seule requête, chaque machine du réseau réflecteur envoie une réponse à la cible. Par conséquent, le débit de l'attaque est multiplié par le nombre de machines

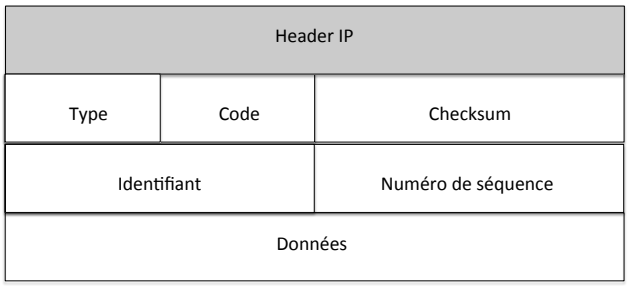

FIGURE 2.5 – Entête ICMP

réflectives.

L'acheminement de paquets de broadcast par le routeur aux membres du réseau était une obligation dans le protocole IPv4 d'après le RFC 1812 [Bak]. Cependant, avec l'émergence des attaques par déni de service, et plus particulièrement, les attaques SMURF, un nouveau RFC 2644 [Sen] a été proposé. Il recommande le blocage par défaut en entrée du réseau de l'acheminement de paquets vers des adresses broadcast. Toutefois, certains réseaux ne respectent pas cette recommandation.

Des réseaux qui acceptent les adresses broadcast sont recensés dans des listes comme celle publiée par Powertech [Pow13]. Dans cette dernière, l'adresse de chaque réseau est accompagnée par le nombre de machines incluses. Par exemple, cette liste inclut un réseau qui accepte les adresses broadcast et qui contient 520 machines. Par conséquent, une requête envoyée à ce réseau générera 520 réponses. Ceci correspond à une amplification de l'attaque SMURF avec un facteur de 520.

L'entête d'un message ICMP écho est décrit dans la figure 2.5. Le champ "Type" permet de différencier les différents messages ICMP. Une valeur de "0" correspond à la requête "echo reply" et la valeur "8" correspond à la réponse "echo reply". Le champ "Code" est négligé dans les messages de type "echo request/reply".

Les champs "Identifiants" et "Numéro de séquence" sont initialisés dans la requête "echo request". Ils sont recopiés dans la réponse et leur valeur reste inchangée. Le champ "Identifiant" permet d'unir les couples "ICMP echo/request" d'une même session puisqu'une application légitime génère des requêtes en rafale. Contrairement au champ "Identifiant", la valeur du champ "Numéro de séquence" est unique pour chaque couple ICMP "ICMP echo/request". En effet, lors de l'exécution de l'application Ping, les différentes requêtes auront le même identifiant, mais un "numéro de séquence" différent.

Enfin, le champ "donnée" ou "Data" est inclus dans les messages de type écho. Sa longueur n'est pas fixe. Elle est définie par l'application qui le génère. Sa valeur n'a généralement pas d'importance pour les applications légitimes. Elle y est fixée de manière aléatoire. Dans les attaques par déni de service ICMP de type saturation, les attaquants génèrent des paquets contenants des champs de données de taille importante pour maximiser l'impact chez la cible.

Dans les approches de traçabilité de type log de paquets, les routeurs enregistrent l'entête du paquet IP ainsi que le premier octet de la charge utile. Ainsi, lors de la détection d'une attaque, la cible demande aux routeurs du réseau de vérifier les paquets qu'ils ont enregistrés afin de déterminer le chemin d'attaque.

Dans une attaque SMURF la cible reçoit un message ICMP "echo reply" de la part de la machine réflective. Si l'approche de traçabilité de type log de paquets est appliquée,

la cible pourra remonter jusqu'à la machine réflective. Ceci est inutile puisque la machine réflective utilise son adresse IP dans la réponse.

Lors de la réception d'une requête, la machine réflective génère un nouveau paquet IP afin d'envoyer la réponse. Par conséquent, les valeurs des champs dits invariables de l'entête IP de la requête et de la réponse sont différentes. Ainsi, la cible ne peut pas remonter à la source de l'attaque puisque les routeurs ont besoin de l'entête IP de la requête alors que la cible ne possède que l'entête de la réponse.

Les approches de marquage de paquet probabiliste ou déterministe utilisent le champ "identifiant de fragment" pour transporter les informations de traçabilité. Puisqu'un nouveau paquet IP est généré pour encapsuler la réponse, les marquages effectués sur l'entête IP par les routeurs entre l'attaquant et les machines réflectives sont perdus.

La cible de l'attaque ne recevra que les marquages des routeurs qui sont entre elle et la machine réflective. Ces marquages lui permettent de reconstruire uniquement le chemin des routeurs entre elle et la machine réflective. Ce qui n'est pas nécessaire puisque les machines réflectives utilisent leur adresse IP réelle lors de l'envoi des réponses vers la cible. En conséquence, dans le marquage de l'entête IP, la cible ne peut pas reconstruire le chemin d'attaque entre les machines réflectives et l'attaquant.

Dans l'approche de traçabilité de type signalisation, la cible d'une attaque reçoit les paquets de signalisation des routeurs qui se situent entre elle et les machines réflectives. Elle ne reçoit pas les paquets de signalisation des routeurs entre les machines réflectives et l'attaquant. En conséquence, la cible ne pourra pas reconstruire le chemin d'attaque entre les machines réflectives et l'attaquant.

Une adaptation de l'approche de signalisation a été proposée par Baros[Bar00] pour la traçabilité des attaques réflectives. Dans cette méthode, les routeurs génèrent des messages de signalisation vers l'émetteur d'un flux contrairement à la méthode de signalisation classique où les routeurs envoient les messages de signalisation vers la destination. En conséquence, dans une attaque réflective c'est la cible qui recevra tous les messages de signalisation.

Comme les routeurs génèrent les messages de signalisation sans tenir compte de la légitimité du flux, ils envoient surtout ces messages vers les émetteurs. En conséquence, lors d'attaques directes, la cible ne recevra pas de paquets signalisation et ne pourra pas reconstruire le chemin de l'attaque.

En plus du surplus de bande passante engendré, l'approche de signalisation pour la traçabilité des attaques réflective a besoin d'un nombre de paquets d'attaques important de l'ordre de quelques milliers afin de reconstruire le chemin entre l'attaquant et la machine réflective.

2.3 Définition, conception d'une méthode de traçabilité

Les attaques par déni de service basées sur les messages ICMP écho peuvent être classées en deux catégories : directes et indirectes. Dans les attaques directes, l'attaquant envoie des paquets en direction de la cible. Les attaques directes peuvent viser à submerger le lien réseau de la cible avec des paquets ICMP écho request ou écho reply. Ces attaques peuvent également viser le système d'exploitation de la cible comme l'attaque Ping of death [Ken96] durant laquelle l'attaquant envoie des paquets ICMP echo qui dépasse la taille maximale autorisée par la couche IP qui est de 65535 octets. Ceci peut provoquer l'arrêt du système d'exploitation.

L'attaque ICMP écho indirecte est appelée SMURF. Elle vise à submerger le lien

réseau de la cible de paquets ICMP écho reply. Pour cela, l'attaquant envoie des requêtes ICMP écho request vers une machine ou un réseau tiers en utilisant l'adresse source de la cible au lieu de la sienne. Ces derniers répondent à la requête en générant des paquets ICMP en direction la cible de l'attaque. Cette redirection permet dans certains cas d'amplifier le débit de l'attaque.

Nous proposons une nouvelle méthode de traçabilité qui permet de remonter à la source d'attaques par déni de service de type ICMP directes et indirectes. Elle permet à la cible de reconstruire le chemin inverse de l'attaque jusqu'à sa source.

Notre méthode de traçabilité est basée sur le marquage de paquets ICMP écho. Les routeurs du réseau insèrent des informations de traçabilité dans les paquets. Afin que le marquage soit efficace pour remonter à la source d'attaques directes et indirectes, les informations de traçabilité doivent être conservées lors de la génération de la réponse par les machines réflectives pour être reçues par la cible.

Le champ "donnée" des paquets ICMP écho n'est pas modifié lors de la génération de la réponse par la machine réflective. Dans le cas d'une attaque réflective, il est initialisé par l'attaquant dans le paquet "ICMP echo request", et sa valeur n'est pas modifiée par la machine réflective dans la réponse "ICMP echo reply" envoyée à la cible. Dans le cas d'une attaque directe, que ce soit les requêtes "echo request" ou réponses "echo reply" qui sont utilisées, la cible recevra également le champ "donnée"initialisé par l'attaquant.

Dans notre approche de traçabilité, les routeurs insèrent leurs informations de traçabilité dans le champ "donnée" des paquets ICMP écho. Puisque ce champ n'est pas modifié par la machine réflective lors de la génération de la réponse, la cible de l'attaque pourra recevoir les informations de traçabilité insérées par les routeurs positionnés entre elle et l'attaquant. En utilisant ce champ de marquage, la cible peut reconstruire le chemin d'une attaque directe ou réflective.

Dans une attaque par déni de service directe, les attaquants peuvent envoyer les réponses "ICMP echo reply" en direction de la cible. Pour pouvoir remonter à la source de ce type d'attaque, les routeurs du chemin dans notre approche insèrent leurs informations de traçabilité dans le champ "donnée" des réponses en plus des requêtes. En conséquence, dans une attaque réflective, la cible pourra reconstruire le chemin complet de l'attaque, de l'attaquant à la machine réflective et de cette dernière à la cible.

Les routeurs insèrent leurs informations de traçabilité dans les paquets ICMP écho. Ces informations sont utilisées par la cible pour remonter à la source de l'attaque :

- attaques par inondation directe : le marquage des différents routeurs du chemin inséré dans le champ donnée des requêtes ou des réponses est reçu par la cible ;
- Ping of Death (PoD) : l'attaquant envoie des messages ICMP écho en direction de la cible. Cette dernière utilise les marquages insérés par les routeurs pour reconstruire le chemin.
- attaques réflectives : la cible reçoit les marquages des routeurs du chemin de l'attaque insérés dans les requêtes et les réponses puisque le champ donnée est recopié par la machine réflective.

Le couple de paquets ICMP écho requêtes/réponses est utilisé dans des applications légitimes comme Ping et Traceroute. L'application Ping envoie des paquets "ICMP echo request" vers un hôte qui lui répond avec des messages "ICMP echo request". Elle est essentiellement utilisée pour tester l'accessibilité d'une machine distante ou pour calculer le temps de l'aller-retour des paquets dans le réseau. L'application Traceroute dans Unix/Mac ou Tracert dans Windows peut envoyer des paquets ICMP écho afin de calculer le nombre de routeurs vers une destination.

Ces deux applications initialisent le champ "donnée" avec une valeur fixe ou aléa-

toire. Le bon fonctionnement de ces applications n'est pas déterminé par la valeur de ce champ. Il est déterminé seulement par les champs de l'entête ICMP du paquet.

Les messages ICMP écho requête/réponse peuvent être utilisés pour monter un tunnel entre deux machines distantes [SNLDS03]. Ce genre de tunnel est utilisé pour contourner certains pare-feux ou pour fausser une communication inaperçue. Ils utilisent le champ "donnée" [SNLDS03] du message ICMP pour transporter le corps de leurs messages. Ils encapsulent les paquets IP dans la charge utile des messages ICMP.

Ce tunnel est utilisé essentiellement par les activités malveillantes [SNLDS03] dans des botnets comme TribeFlood Net 2k et Stacheldrah pour faire communiquer les machines infectées avec leur serveur de contrôle. Ces botnets sont utilisés pour lancer des attaques par déni de service distribuées.

Dans notre méthode de marquage, les routeurs insèrent leurs informations de traçabilité dans le champ "donnée" des messages ICMP écho. La taille de ce champ est plus importante que celle du champ "identifiant de fragment" utilisé dans les méthodes de marquage IP. Dans notre méthode de marquage, un seul paquet peut transporter les informations de traçabilité de plusieurs routeurs alors qu'il faut plusieurs paquets pour transporter les informations de traçabilité d'un seul routeur dans les approches de marquage IP. Ceci permet de réduire le nombre de paquets nécessaires à la cible pour reconstruire le chemin de l'attaque.

Les informations de marquage insérées par les routeurs du chemin permettent à la destination de reconstruire le chemin inverse du flux afin d'identifier la source de celui-ci. Ces informations incluent un identifiant qui peut être l'adresse IP du routeur ou toute autre valeur pouvant l'identifier, s'il ne peut pas divulguer son adresse IP. Cet identifiant peut être constitué d'une valeur attribuée par l'opérateur suivi du numéro de l'AS (Système Autonome).

Dans notre approche, les routeurs n'utilisent pas la totalité du champ "donnée" pour insérer leurs informations de traçabilité. Ils utilisent seulement une partie. Ceci pour lutter contre l'usurpation de marquage effectuée par l'attaquant. Ce dernier peut faire de l'usurpation de marquage en insérant de fausses informations de marquage dans les paquets d'attaques, s'il est au courant de la méthode de traçabilité implémenté.

Dans l'usurpation de marquage, l'attaquant utilise le format de notre méthode de traçabilité dans le champ "donnée" des paquets ICMP écho générés. Il insère dans ce champ de fausses informations de marquages de routeurs afin d'abuser la cible sur le chemin pris par les parquets.

Tous les routeurs du chemin ne peuvent pas insérer leur information de traçabilité dans le champ de marquage. Sachant qu'un paquet transite en moyenne par 15 routeurs [KBMA+09] avant d'atteindre sa destination sachant que cette moyenne est doublée dans le cas d'une attaque réflective. Ainsi, pour permettre à tous les routeurs du chemin d'insérer leurs informations dans le champ de marquage, nous utilisons une approche probabiliste. Les routeurs insèrent leurs informations dans le champ de marquage suivant une probabilité p.

Puisque le champ de marquage ne permet pas de contenir toutes les informations de traçabilité du chemin, un routeur peut ne pas trouver d'espace disponible pour insérer ses informations de marquage. Ceci à cause des routeurs précédents ou à cause de l'attaquant qui a rempli l'espace de marquage avec de fausses informations de traçabilité. Ainsi, si les routeurs ne peuvent pas marquer le message, la cible de l'attaque peut recevoir uniquement de fausses informations de traçabilité et aucune information sur les routeurs du chemin. En conséquence, dans notre méthode de traçabilité, si un routeur décide de marquer un message et qu'il ne trouve pas d'espace disponible, il écrase un

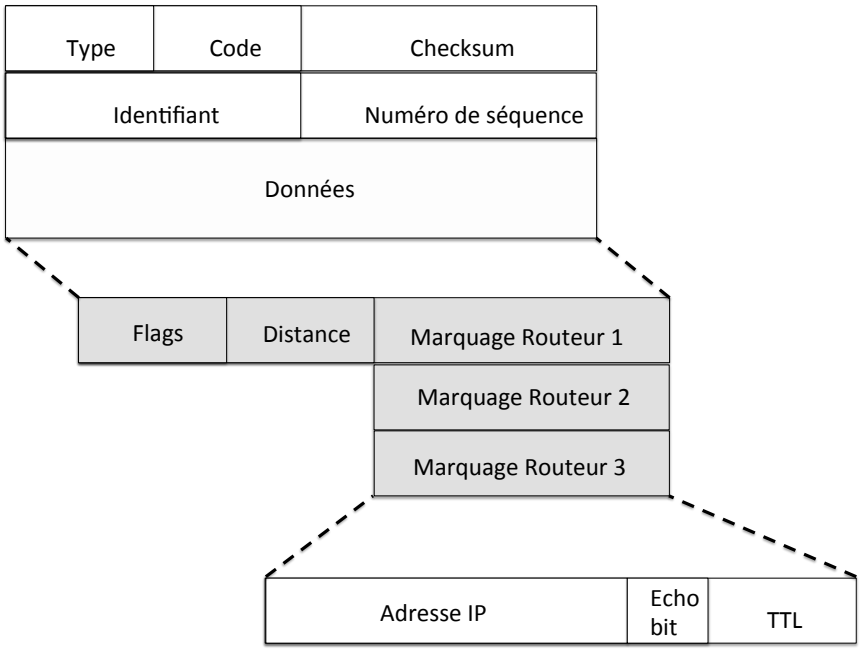

FIGURE 2.6 – Marquage du champ "donnée" d'un message ICMP écho

marquage précédent.

Nous définissons la structure de l'espace de marquage utilisé dans notre approche de traçabilité dans la figure 2.6. Il contient un champ Flag, un champ distance, et des champs contenants les informations de traçabilité des différents routeurs du chemin. Ces informations de traçabilité comportent l'adresse IP du routeur, un bit écho, et le TTL de l'entête IP.

Avant d'insérer leurs informations de marquage, les routeurs doivent avoir la possibilité de vérifier si le paquet a été marqué auparavant. La cible d'une attaque doit aussi avoir la possibilité de le vérifier avant de récupérer les informations de marquage. Pour cela, ils vérifient la présence du "Flag" au début du champ "donnée".

Les routeurs insèrent leurs informations de traçabilité dans l'espace de marquage. Pour que la cible de l'attaque reconnaisse les routeurs du chemin, ils doivent insérer leur adresse IP. De plus, pour que la cible puisse situer les routeurs dans le chemin de l'attaque, ces derniers insèrent la valeur du champ TTL de l'entête du paquet IP lors du marquage. La cible compare les TTL des différents marquages pour situer la position des routeurs dans le chemin de l'attaque. En effet, le champ TTL de l'entête IP est décrémenté par chaque routeur. Par conséquent, plus la valeur du champ TTL d'un marquage est grande, plus il s'approche de celui qui a généré l'entête IP, l'attaquant ou la machine réflective.

Dans une attaque réflective, la cible reçoit les marquages des routeurs positionnés entre elle et la machine réflective et entre cette dernière et l'attaquant, puisque les routeurs insèrent leurs informations de traçabilité dans les requêtes et les réponses. Pour que la cible puisse différencier ces routeurs et reconstruire le chemin complet de l'attaque, ils ajoutent dans le champ de marquage un "bit écho" afin de différencier les requêtes et les réponses.

Algorithm 1 Algorithme de marquage

p_i probabilité de marquage
$Mark_{max}$ Nombres de marquages maximums
Pour chaque message ICMP écho P
$xréel \in [0, 1]$ ▷ choisi aléatoirement
$Position$ Entier
if $x < p_i$ **then**
    if P n'est pas marqué **then**
        initialiser le champ Mark.Flag
        $Mark.Distance \leftarrow 0$
        $Position \leftarrow 0$ ▷ Marquer dans la première position
    else
        if $Mark.Distance \geq Mark_{max}$ **then** ▷ L'espace de marquage est rempli
            $Position \in [0, Mark_{max} - 1]$ ▷ Valeur choisie aléatoirement
        else
            $Position \leftarrow Mark.Distance$ ▷ Valeur choisie aléatoirement
        end if
    end if
    $Mark.Distance \leftarrow Mark.Distance + 1$ ▷ Incrémenter la distance
    $Mark.Marquage[Position].IP \leftarrow Routeur.IP_{address}$
    $Mark.Marquage[Position].TTL \leftarrow P.TTL$
    if P echo request **then**
        $Mark.Marquage[Position].echo \leftarrow 1$
    else
        $Mark.Marquage[Position].echo \leftarrow 0$ ▷ Si c'est un echo reply
    end if
end if

L'algorithme de marquage de notre méthode de traçabilité est décrit dans l'algorithme 1. Les routeurs insèrent leurs informations de traçabilité dans les messages ICMP écho avec une probabilité prédéfinie. Lorsqu'ils acheminent un paquet, s'il décide de le marquer, ils vérifient s'il a été marqué auparavant. Pour cela, il vérifie la présence du Flag au début du champ "donnée". Si le paquet n'a pas été marqué auparavant, le routeur initialise le Flag pour informer les prochains routeurs et la cible de la présence d'un marquage. Afin de prévenir les prochains routeurs de la présence d'un seul marquage, il initialise aussi la distance à 1 .

L'espace de marquage est constitué d'emplacements pouvant accueillir les informations de traçabilité de plusieurs routeurs. Chaque emplacement contient les informations de traçabilité d'un routeur du chemin. Le nombre d'emplacements de marquage est limité par la taille de l'espace total de marquage.

Le routeur choisit un emplacement par rapport à la distance. Si cette distance est inférieure au nombre d'emplacement admis, le routeur marquera ses informations dans l'emplacement disponible suivant correspondant à la valeur de cette distance. Par contre, si la valeur de la distance est supérieure ou égale au nombre d'emplacements admis, c'est-à-dire que tous les emplacements ont été utilisés, le routeur écrase un de ces emplacements pour insérer ses informations. Il choisit cet emplacement de manière aléatoire.

Les routeurs insèrent le TTL de l'entête IP du paquet pour permettre à la cible de l'attaque de les situer dans le chemin de l'attaque. Dans le cas d'une attaque directe, plus la valeur du TTL est grande, plus le routeur est proche de l'attaquant puisque sa valeur est décrémentée lors du routage. Par contre, dans une attaque réflective la cible distingue les marquages des requêtes et des réponses :

- Dans les requêtes, plus la valeur du TTL est élevée, plus le routeur est proche de l'attaquant.
- Dans les réponses, plus la valeur du TTL est élevée, plus le routeur est proche de la machine réflective.

Afin de limiter les conséquences de l'usurpation de marquage, les routeurs vérifient les champs TTL marqués dans le même message, voir algorithm 2. Si le message acheminé est une requête, le routeur vérifie la valeur du TTL des champs de marquage effectués auparavant. Sinon, il vérifie la valeur du TTL des champs de marquage de réponses uniquement. Si le TTL marqué est inférieur au TTL de l'entête du paquet IP lors de l'acheminement du paquet, le routeur efface le marquage puisqu'il doit être supérieur au TTL de l'entête. Il est inférieur parce qu'il est décrémenté par tout les routeurs du chemin. En conséquence, un TTL marqué inférieur au TTL de l'entête du paquet IP correspond forcément à un faux marquage inséré par l'attaquant pour perturber la reconstruction du chemin de l'attaque.

2.4 Évaluation et validation

Afin de valider notre méthode de traçabilité du trafic IP, nous avons implémenté les différents algorithmes nécessaires à son bon fonctionnement : l'algorithme de marquage des messages ICMP écho, l'algorithme de reconstruction du chemin de l'attaque, le programme d'attaque ICMP directe et indirecte.

Nous nous sommes basés sur un routeur Linux (Debian) exécutant notre programme de marquage pour valider notre méthode de traçabilité. Ce programme insère de manière probabiliste les informations de traçabilité du routeur (adresse IP, type de ICMP écho et

Algorithm 2 Algorithme de marquage avec vérification d'usurpation

p_i probabilité de marquage
$Mark_{max}$ Nombres de marquages maximums
Pour chaque message ICMP écho P
$xréel \in [0, 1]$ ▷ choisi aléatoirement
$Position$ Entier
if $x < p_i$ **then**
  if P n'est pas marqué **then**
    initialiser le champ Mark.Flag
    $Mark.Distance \leftarrow 0$
    $Position \leftarrow 0$ ▷ Marquer dans la première position
  else
    if $Mark.Distance \geq Mark_{max}$ **then** ▷ L'espace de marquage est remplis
      $Position \in [0, Mark_{max} - 1]$ ▷ Valeur choisie aléatoirement
      $Mark.Distance \leftarrow Mark_{max} - 1$ ▷ Valeur maximale de la distance avant incrémentation
    else
      $Position \leftarrow Mark.Distance$ ▷ Valeur choisie aléatoirement
    end if
  end if
  $Mark.Distance \leftarrow Mark.Distance + 1$ ▷ Incrémenter la distance
  $Mark.Marquage[Position].IP \leftarrow Routeur.IP_{address}$
  $Mark.Marquage[Position].TTL \leftarrow P.TTL$
  if P echo request **then**
    $Mark.Marquage[Position].echo \leftarrow 1$
  else
    $Mark.Marquage[Position].echo \leftarrow 0$ ▷ Si c'est un écho reply
  end if
  for $i \leftarrow 1, Mark.Distance$ **do** ▷ Vérifier la présence de marquage usurper
    if $Mark.Marquage[i-1].echo = Mark.Marquage[Position].echo$ **then**
      if $P.TTL \geq Mark.Marquage[i-1].TTL$ **then**
        effacer le marquage Mark.Marquage[i - 1]
      end if
    end if
  end for
end if

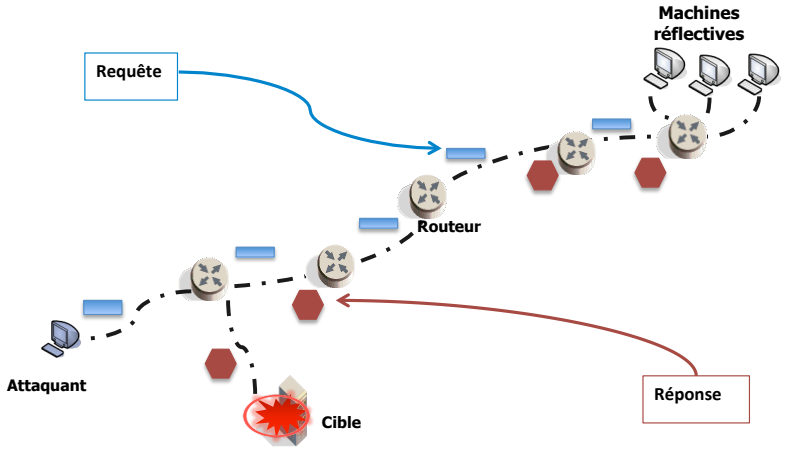

FIGURE 2.7 – Scénario de l'évaluation de notre méthode de marquage

TTL de l'entête IP). Nous avons configuré le pare-feu du système pour qu'il renvoie les messages ICMP écho vers notre programme de marquage, qui à son tour, les renvoie vers l'interface de sortie après traitement.

Nous avons implémenté notre algorithme de reconstruction du chemin de l'attaque au niveau de la cible. Comme dans les routeurs, nous avons configuré le pare-feu du système pour rediriger les messages "ICMP echo" à leur réception vers notre programme de reconstruction du chemin de l'attaque. Ce programme extrait les informations de traçabilité des messages.

Nous nous sommes basées sur une autre machine Linux (Debian) pour exécuter le programme d'attaque ICMP. Ce programme peut lancer des attaques directes ou réflectives durant lesquelles il peut usurper l'adresse IP source des paquets. Dans l'attaque réflective, le programme envoie des paquets en direction de machines réflectives en utilisant comme adresse source celle de la cible. Ce programme permet également de configurer la taille du champ donnée des paquets d'attaques.

Pour les machines réflectives, nous avons utilisé des machines Linux bien configurées. Le rôle de ces machines dans ce scénario est de répondre aux requêtes ICMP écho.

Nous avons mis en place, dans un réseau virtuel fermé, les différents acteurs de la traçabilité d'une attaque par déni de service ICMP indirecte : l'attaquant, la cible, cinq routeurs, et trois machines réflectives comme illustré dans la figure 2.7. Nous avons organisé les routeurs pour qu'ils soient traversés par les requêtes et les réponses de l'attaque réflective.

Pour évaluer notre méthode de marquage, nous avons lancé l'attaque SMURF en utilisant 3 machines réflectives. Nous avons fixé la probabilité de marquage à 1 pour les différents routeurs du chemin, et l'espace de marquage à 100 octets, afin de permettre à tous les routeurs du chemin d'insérer leurs informations de traçabilité. Par ailleurs, pour acheminer l'espace de marquage défini, nous avons utilisé durant l'attaque des messages ICMP écho de grande taille, dépassant 1000 octets, voir la figure 2.8.

En analysant les paquets d'attaques reçus par la cible, nous avons remarqué qu'elle a reçu les informations de traçabilité de tous les routeurs du chemin, voir la figure 2.9. La cible a pu extraire les différents marquages à partir du paquet d'attaques afin de

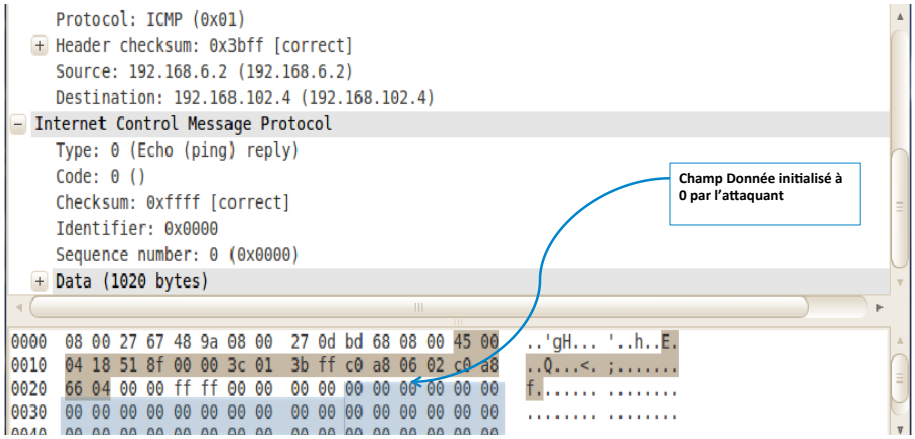

FIGURE 2.8 – Capture d'écran d'un message ICMP écho reply non marqué

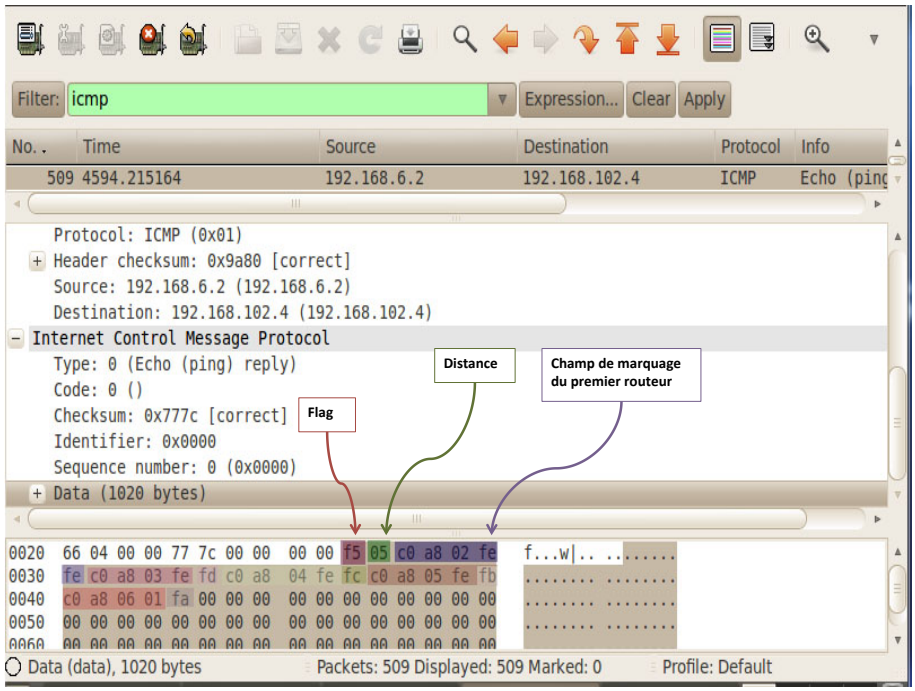

FIGURE 2.9 – Capture d'écran d'un message ICMP écho reply marqué par 5 routeurs

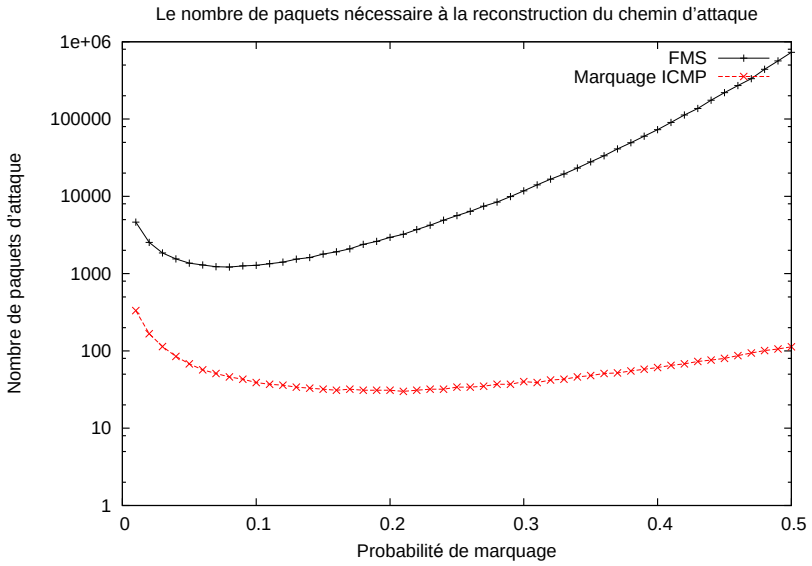

FIGURE 2.10 – Nombre de paquets de reconstruction par rapport à la probabilité de marquage de la méthode FMS et de la méthode de marquage ICMP

reconstruire le chemin de l'attaque. Elle s'est basée sur la valeur des champs TTL accompagnant les adresses IP pour discerner la position des routeurs dans le chemin.

Notre méthode de traçabilité permet de remonter à la source des attaques par déni de service directe et indirecte. Ainsi, pour l'évaluer, nous l'avons comparé avec les méthodes de marquage probabilistes de l'entête IP lors d'une attaque directe. Nous avons calculé le nombre de paquets minimum pour reconstruire le chemin d'une attaque ICMP écho directe.

Nous avons calculé le nombre de paquets minimum pour reconstruire le chemin d'une attaque dans notre méthode de traçabilité et dans la méthode FMS par rapport à la probabilité de marquage des routeurs, voir la figure 2.10. On remarque que le nombre de paquets nécessaires est largement inférieur dans notre méthode quelques soit la probabilité de marquage considéré. En effet, notre méthode de marquage se base sur un espace de marquage plus important.

Dans notre approche, nous nous sommes basés sur un faible espace de marquage de 12 octets pour compliquer l'usurpation de marquage et pour s'intégrer dans les applications légitimes. Cet espace permet de transporter les informations de traçabilité de 2 routeurs dans un seul message. Nous avons fixé le nombre de routeurs entre l'attaquant et la victime à 15 routeurs puisque c'est la distance moyenne entre deux hôtes dans réseau Internet [KBMA+09].

Afin de déterminer la probabilité de marquage optimale de notre méthode de traçabilité, nous avons calculé le nombre de paquets nécessaires pour reconstruire le chemin d'attaque dans le cas d'une attaque directe et d'une attaque réflective, voir la figure 2.11. Nous avons fixé le nombre de routeurs dans l'attaque directe à 15 routeurs et à 30 routeurs dans le cas de l'attaque indirecte. On remarque que le nombre de paquets nécessaire dans les deux attaques décroît lorsque la probabilité est inférieure à 0.1 et il croit au-dessus de cette valeur dans le cas de l'attaque réflective. Par contre, il est à peu

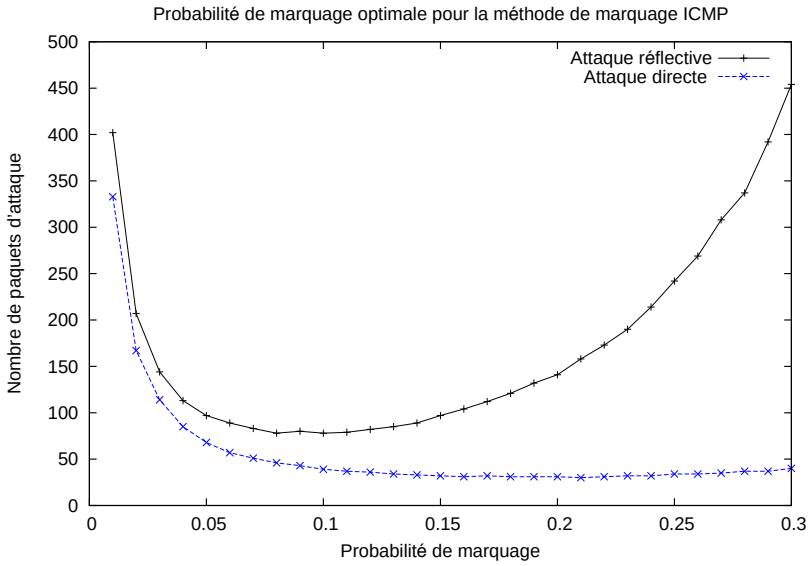

FIGURE 2.11 – Nombre de paquets de reconstruction par rapport à la probabilité de marquage d'une attaque et d'une attaque réflective

près stable dans le cas d'une attaque directe.

Lorsque la probabilité de marquage dépasse 0.1, le nombre de paquets de reconstruction est croissant à cause de l'écrasement de marquage. Les routeurs proches de la cible de l'attaque écrasent les marquages de ceux qui sont proches de l'attaquant. En conséquence, la probabilité optimale de marquage est 0.1

Nous avons aussi fait varier le nombre de paquets d'attaques nécessaire à la reconstruction par rapport au chemin pour la méthode de marquage FMS, notre méthode de marquage ICMP, et la signalisation, voir la figure 2.12. Nous avons fixé la probabilité de marquage à 0.1 pour notre méthode et de 0.04 pour le marquage FMS. On remarque que le nombre de paquets nécessaire pour reconstruire le chemin d'attaque s'accroit avec le nombre de routeurs dans les deux méthodes de marquage. Par contre, il est stable pour signalisation. On remarque aussi que notre méthode de marquage nécessite un nombre de paquets d'attaques largement inférieur aux autres méthodes quelques soit la longueur du chemin.

2.5 Conclusion et analyse

Dans ce chapitre, nous avons proposé une méthode de traçabilité du trafic qui permet de remonter à la source d'attaques ICMP directes ou réflective. Elle est basée sur le marquage probabiliste des messages ICMP acheminés par les routeurs du réseau. Elle nécessite un nombre de paquets pour la reconstruction dans une attaque directe largement inférieure aux méthodes de marquage et de signalisation.

Notre méthode de traçabilité obéit aux exigences définies par Belenky et al. [BA03a] : elle fonctionne même si elle est implémentée partiellement dans le réseau ; elle passe à l'échelle vu qu'elle fonctionne quelle que soit la taille du chemin ; elle ne génère pas

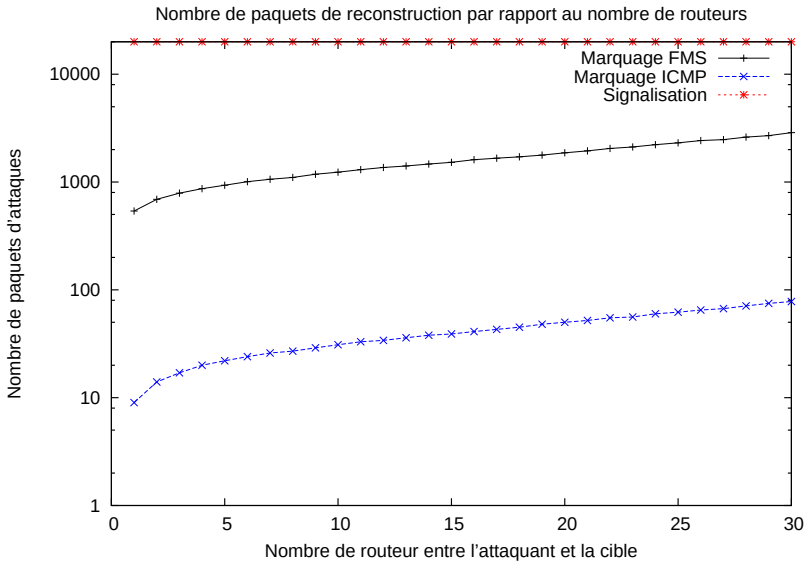

FIGURE 2.12 – Nombre de paquets de reconstruction par rapport au nombre de routeurs

de surplus pour la bande passante puisqu'elle utilise les champs du paquet ; elle peut construire le chemin d'une attaque avec un nombre de paquets largement inférieur aux autres méthodes.

Notre approche est basée sur l'utilisation du champ "donnée" du message ICMP écho pour transporter les informations de traçabilité des routeurs du chemin vers la cible. Le marquage de ce champ peut générer des avertissements lors de l'utilisation d'applications légitimes telles que Ping et Traceroute. Cet avertissement est généré par certaines implémentations qui vérifient le contenu de ce champ.

La traçabilité permet de filtrer le trafic malveillant au plus près des attaquants. Ceci permet aux membres du réseau de la cible d'avoir toujours accès au service proposé par cette dernière lorsqu'elle est sous le coup d'une attaque par déni de service. De plus, les membres des réseaux partenaires avec celui de la cible peuvent aussi avoir accès à son service s'ils appliquent le même filtrage au bord de leurs réseaux.

Une attaque par déni de service distribuée est générée par un nombre de machines infectées (bots) sous le contrôle d'un attaquant par l'intermédiaire d'un serveur de contrôle. Une méthode de traçabilité IP permet de remonter aux machines infectées, bien qu'elle soit considérée comme une méthode de dissuasion, elle ne permet pas prévenir une attaque.

Une méthode de détection du serveur de botnet qui contrôle les machines d'attaques permet de prévenir cette dernière. En effet, le blocage de ce serveur empêche les commandes d'attaques d'être reçues par les bots. Ainsi, le réseau de la cible n'aura pas besoin de filtrer les paquets puisque l'attaque ne sera pas enclenchée.

Chapitre 3

Méthode de détection de botnets

3.1 Problématique

Les logiciels malveillants déguisent leurs communications réseau avec leur serveur de contrôle afin de ne pas éveiller les soupçons des systèmes de détection d'intrusion (IDS) au niveau du réseau local. Ces derniers se basent sur les signatures du trafic généré par les botnets pour identifier les communications malveillantes. En revanche, un système de détection au niveau d'un opérateur réseau a la possibilité d'analyser et de corréler le trafic des membres de son réseau afin d'identifier les groupes d'utilisateurs participants aux mêmes activités malveillantes.

Un système de détection de botnet au niveau d'un opérateur doit avoir la capacité d'analyser en temps réel le trafic réseau de millions d'utilisateurs puisque la législation de protection de données d'utilisateurs de plusieurs pays européens[KDG+12b] interdit la conservation du trafic internet des clients. De plus, la conservation des traces de connexion nécessite une mémoire importante à cause de la hausse du débit des utilisateurs grands publics.

La prise de conscience des utilisateurs d'internet de la nécessité de la protection de leur vie privée, les rend méfiants envers les solutions nécessitants une collecte de leurs données personnelles. Cette méfiance inclut également les solutions de sécurité qui analyse leur trafic réseau pour détecter des activités malveillantes.

La combinaison de ces contraintes législatives, techniques et sociales impose à un système de détection d'activité malveillante déployée au niveau d'un opérateur réseau :

- l'anonymisation du trafic des utilisateurs pour protéger leur vie privée,
- et l'analyse en temps réel du trafic réseau à large échelle.

L'objectif d'un système de détection de botnet au niveau d'un opérateur est d'identifier le trafic malveillant généré par les machines infectées pour communiquer avec leur serveur de contrôle. Cette détection est compliquée pour trois raisons principales [SSPS12] :

- Le trafic de contrôle d'un botnet, c'est-à-dire le trafic généré par les machines infectées pour communiquer avec leur serveur de contrôle et commande (C&C), ressemble au trafic légitime. Les botnets se basent sur des protocoles populaires pour communiquer.
- Les botnets évoluent rapidement. Les créateurs de botnets adaptent leurs attaques et leurs méthodes de communication aux nouveaux usages d'internet. Ils s'adaptent aussi aux méthodes de détection ou de protection mise en oeuvre.
- La quantité importante de données à analyser afin d'identifier le trafic malveillant.

Le point commun entre les différents botnets est la communication des machines infectées avec leur serveur de contrôle et commande (C&C), afin de recevoir des com-

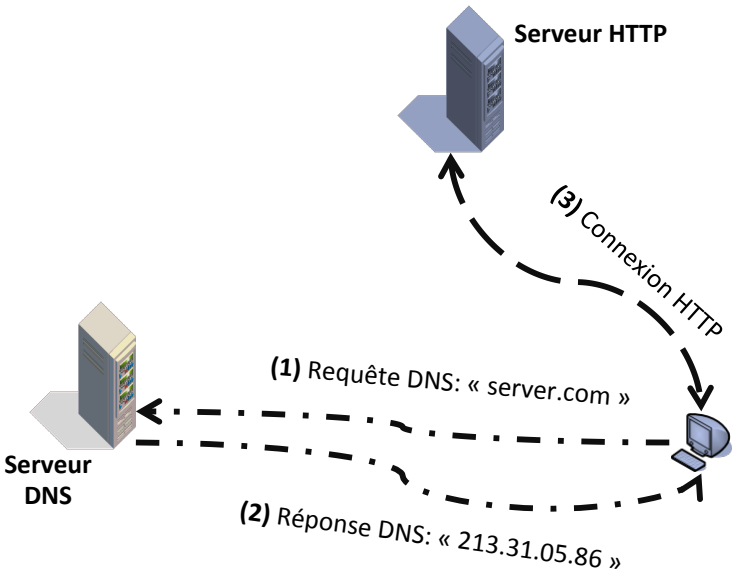

FIGURE 3.1 – Connexion à un serveur web

mandes ou pour envoyer les rapports d'attaques en cours. En conséquence, une méthode générique de détection de botnet doit pouvoir identifier le canal de communication des machines infectées avec leur serveur de contrôle, quelles que soient les attaques générées.

L'identification du canal de contrôle du botnet permet de mettre en oeuvre des contre-mesures, afin de prévenir les attaques générées. Une contre-mesure au niveau des communications du botnet est plus efficace qu'une contre-mesure appliquée pour atténuer l'impact d'une attaque réseau en cours puisqu'elle permet d'intervenir avant sont apparition. Par exemple, l'identification et le blocage du canal de contrôle d'un botnet spécialisé dans les attaques par déni de service distribuées permet d'éviter l'attaque DDoS en empêchant les commandes d'arriver aux machines infectées.

Les botnets utilisent des protocoles réseau usuels, principalement HTTP et IRC, pour l'interaction entre les logiciels malveillants et leur serveur de contrôle et commande (C&C). L'analyse de ces protocoles au niveau d'un opérateur réseau est difficile à cause de leur proportion dans le trafic internet. L'utilisation du chiffrement au-dessus de ces protocoles complique encore plus leur détection.

Pour se connecter au serveur de contrôle et commande, les machines infectées ont besoin de son adresse IP. Afin de récupérer cette adresse, les botnets utilisent le protocole DNS (Domain Name System). Ceci leur permet de rendre architecture du botnet plus robuste par rapport à l'utilisation d'adresses IP fixes. En plus de l'utilisation classique du protocole DNS, c'est-à-dire l'association entre un nom de domaine et une ou plusieurs adresses IP, les opérateurs de botnets utilisent des techniques tels que le fast-flux [NH08] et le domain-flux [SGCC+09] afin de les rendre plus résistant aux tentatives de blocages.

Pour se connecter à un serveur web, un utilisateur muni de son nom de domaine envoie une requête au serveur DNS afin de récupérer son adresse IP comme illustrée dans la figure 3.1. Une fois cette adresse obtenue, l'utilisateur enclenche la communication

avec le serveur web. Ceci implique dans ce cas l'établissement d'une connexion HTTP avec le serveur.

En plus de l'adresse IP du serveur, la réponse DNS contient le champ TTL (Time To Live). La valeur de ce champ correspond à la durée pour laquelle l'association entre l'adresse IP et le nom de domaine est valide. Tant que cette association est valide, l'utilisateur communique avec l'adresse IP du serveur et ne génère pas de nouvelle requête DNS. En effet, si la valeur du TTL d'un nom de domaine est de 600 secondes, la machine de l'utilisateur garde l'association entre le nom de domaine et l'adresse IP dans sa mémoire, et ne génère pas de nouvelles requêtes DNS vers ce nom de domaine avant l'écoulement du TTL. En conséquence, le protocole DNS représente qu'une faible partie du trafic réseau généré lors des communications. Il représente moins de 1% [SM09] du trafic réseau global.

L'utilisation du protocole DNS par la majorité des botnets et son faible ratio dans le trafic réseau global rend les méthodes de détection dans les réseaux à large échelle basées sur ce protocole attractives. Puisque les utilisateurs se basent en majorité sur les serveurs DNS de l'opérateur pour la résolution des requêtes, la capture du trafic DNS à la hauteur du serveur DNS de l'opérateur permet de recevoir la majorité du trafic DNS. Le trafic DNS restant peut être capturé dans les routeurs de bord du réseau.

Les méthodes de détection de botnets basées sur le protocole DNS permettent d'identifier les noms de domaine malveillants utilisés par les botnets ainsi que les adresses IP des serveurs de contrôle. Le blocage des noms de domaine malveillants est très utilisé pour lutter contre les botnets. Les machines infectées ne peuvent pas récupérer l'adresse IP de leur serveur de contrôle. Ainsi, elles ne peuvent pas se connecter au serveur. Par conséquent, le botnet devient inopérable pour ses opérateurs.

Pour contourner ce blocage, les opérateurs de botnets utilisent une technique appelée domain-flux. Cette technique permet aux membres du botnet de générer différents points de rendez-vous. L'opérateur du botnet connait ces points de rendez-vous au préalable, et en choisit un seul afin de l'associer au serveur de contrôle et commande de son botnet. Ces points de rendez-vous sont des noms de domaine générés de manière algorithmique par les logiciels malveillants du botnet. Le module responsable de cette génération est appelé DGA (Domain Generation Algorithm).

Le module de génération de noms de domaine (DGA) permet aux machines infectées distribuées géographiquement de générer les mêmes noms de domaines. Ils s'assurent qu'ils génèrent la même liste, quel que soit leur emplacement. Ils utilisent des paramètres accessibles à toutes machines infectées en entrée de l'algorithme de génération de noms de domaine telles que l'heure actuelle ou le contenu d'une page web particulière.

Sur la liste des noms de domaine générés par les membres du botnet de type domain-flux, quelques-uns seulement sont réservés et utilisés pour le serveur de contrôle. Le reste des noms de domaine ne sont pas réservés et les requêtes DNS envoyées par les machines infectées afin de les résoudre produiront des réponses DNS spécifiques de type domaine inexistant (NXDomain).

La technique de blocage de noms de domaine malveillants est difficilement applicable aux botnets de type domain-flux. Pour que la technique soit effective, il faudrait bloquer tous les noms de domaine utilisés par le botnet. Ce blocage devrait être renouvelé chaque fois que le botnet génère une nouvelle liste de noms de domaine. Ce qui peut atteindre 50000 noms de domaine par jour, dans le cas du botnet Conficker.C [LW09].

Ce blocage, pour être efficace, devrait ajouter les noms de domaine malveillants avant leur génération, afin d'empêcher tous les membres du botnet de se connecter à leur serveur de contrôle. Pour anticiper les noms de domaine qui seront générés par les

machines infectées, il faudrait faire de l'ingénierie inversée sur le logiciel malveillant du botnet afin d'élucider l'algorithme de génération de noms de domaine.

La technique domain-flux est utilisée par plusieurs botnets : Conficker et ses différentes variantes [LW09], Torpig [SGCC+09], Mega-D [Lin09], Kraken [Roy08], PushDo [Dam13] et quelques versions de Zeus [Zeub].

3.2 Analyse des méthodes de détection existantes

La détection de botnets se fait essentiellement au niveau du système de la victime en utilisant des solutions classiques de détection comme les antivirus. Même si ces solutions détectent et évitent l'infection du système de beaucoup de logiciels malveillants, ils ne détectent pas toutes les menaces. En effet, selon le site d'analyse de fichiers Virustotal [Vir13], il n'existe pas de solution garantissant la détection de 100% des logiciels malveillants. Par conséquent, une machine protégée par un antivirus à jour est toujours vulnérable à certaines attaques.

Selon Microsoft [ABB13], 24% des machines connectées à internet n'ont pas de protection contre les logiciels malveillants parce qu'ils n'ont pas d'antivirus ou parce que ce dernier n'est pas à jour. Ils affirment qu'une machine non protégée est 5.5 fois plus vulnérable aux menaces des logiciels malveillants qu'une machine protégée.

Afin de pallier les limites des solutions de détection au niveau du système et puisque les membres de botnets utilisent le réseau internet pour communiquer avec leur maitre ou pour lancer leurs attaques, plusieurs méthodes de détection de botnet au niveau du réseau ont été proposées. Ces dernières analysent le trafic réseau généré par les machines afin d'identifier le trafic malveillant.

Les systèmes de détection réseau de botnets peuvent être classés en deux catégories : actifs et passifs.

3.2.1 systèmes actifs de type honeypots

C'est un ensemble constitué d'un ou de plusieurs systèmes vulnérables aux infections [Spi03]. Ils se laissent infecter ou exécutent volontairement des logiciels malveillants afin d'analyser leur comportement. Ils permettent aussi d'identifier le serveur de contrôle et commande (C&C) ainsi que les autres composants du botnet.

Les honeypots peuvent être utilisés pour générer des signatures pour des systèmes de détection passifs. Wurzinger et al. [WBH+09] proposent un système de génération de signatures de trafic réseau malveillant pouvant être utilisé ultérieurement par un IDS pour détecter les communications des membres d'un botnet. Ces signatures sont générées en observant la communication des machines infectées du honeypot avec leur serveur de contrôle ainsi que leur comportement ou le trafic réseau généré après la réception d'une commande du serveur.

Le système peut générer des signatures du trafic réseau malveillant uniquement pour les botnets qui ne chiffrent pas leur canal de contrôle et qui sont réactifs à la réception d'une commande. La réaction d'un bot doit être identifiable au niveau de son trafic réseau. Nous pouvons citer comme réaction identifiable l'envoi de spam ou la participation à des attaques par déni de service. Si le botnet n'est pas réactif, le système peut ne pas associer des attaques aux commandes reçues par les machines infectées et ainsi ne pas générer de signatures.

Les honeypots sont exécutés généralement dans des machines virtuelles afin d'optimiser les ressources du système et pour exécuter plusieurs systèmes d'exploitation dans

la même machine physique. Par conséquent, les créateurs de botnets ont ajouté des mécanismes dans leurs logiciels malveillants pour identifier les machines virtuelles [LW09]. Ils peuvent stopper l'exécution de leur logiciel malveillant ou modifier son comportement pour altérer son analyse par le honeypot.

3.2.2 Systèmes passifs de détection de botnets

Ces systèmes analysent le trafic réseau afin d'identifier les machines infectées et leur serveur de C&C. Ils ne génèrent pas de trafic réseau supplémentaire et n'interagissent pas avec les membres d'un botnet. En conséquence, ils sont transparents à l'opérateur du botnet, c'est-à-dire qu'il ne peut pas découvrir leur déploiement dans un réseau.

Plusieurs méthodes passives de détection réseau de botnet ont été proposées ces dernières années. Bothunter [GPY+07] analyse le trafic généré par un réseau local afin d'identifier les machines infectées. Il peut même être intégré au système de détection d'intrusion Snort. Il détecte les bots ou machines infectées en se basant sur une séquence d'événements qui permet de qualifier un comportement comme étant malveillant. Ces événements sont : (1) le scan ou le balayage du réseau de l'extérieur vers le réseau local ; (2) l'injection de code de l'extérieur vers le réseau local ; (3) le téléchargement d'un fichier binaire infecté ; (4) la communication avec le canal de contrôle ; (5) et le scan de l'intérieur du réseau local vers l'extérieur.

Botsniffer est une amélioration de Bothunter [GZL08]. Ce système détecte les botnets centralisés ainsi que leur serveur de contrôle et commande basé sur les protocoles IRC ou HTTP. Ils basent leur système de détection sur la synchronisation et la ressemblance du trafic réseau des membres d'un même botnet. Ils supposent que les membres d'un botnet répondent en même temps à une requête de leur serveur de C&C et qu'ils se connectent périodiquement dessus en utilisant la même période. Ils supposent aussi que les messages envoyés et reçus par les membres d'un même botnet à leur serveur sont homogènes.

Contrairement à Botsniffer qui se concentre sur la détection des botnets centralisés, Botminer [GPZ+08] permet la détection de structures centralisée et décentralisée. Ce système corrèle les communications des machines afin de détecter les connexions similaires ainsi que le comportement malveillant des utilisateurs (scan, envoi de spams, exploits, téléchargement de fichiers infectés ...).

Ces méthodes de détection se basent sur des signatures afin d'identifier les fichiers binaires infectés, les exploits, et les canaux de contrôle de botnets. Elles identifient uniquement les botnets pour lesquels elles possèdent des signatures. Elles ne peuvent pas détecter de nouveaux botnets ou ceux qui utilisent des mécanismes pour faire varier la signature de leurs fichiers binaires. L'identification de l'injection de code ainsi que les fichiers binaires infectés nécessitent une analyse minutieuse du trafic réseau des utilisateurs. Par conséquent, ces approches ne passent pas à l'échelle d'un réseau d'opérateur.

Afin de passer à l'échelle, Zhang et al. [ZLP+11] proposent une méthode de détection de membres d'un même botnet basée sur l'analyse de flux réseau échantillonnés des protocoles TCP et UDP. Cette méthode adapte l'échantillonnage afin de maximiser la capture du trafic de contrôle. Ils utilisent ensuite une méthode spatio-temporelle pour identifier les machines suspectées d'être infectées. Le trafic réseau de ces machines est ensuite analysé de manière plus approfondie pour détecter les machines infectées et leur serveur de contrôle.

Les méthodes précédentes (Botsniffer, Botminer, et la méthode proposée par Zhang et al.) filtrent les connexions vers des sites dits populaires, afin de réduire le volume du

trafic réseau à analyser. Ils se basent sur des listes blanches (Whitelists) de sites web populaires comme celle proposée par le site Alexa [ale]. Ils excluent le trafic de ces sites de leur analyse puisqu'ils les considèrent comme bien entretenu et n'hébergeant pas de contenu malveillant.

Plusieurs approches ont été proposées afin d'identifier les botnets de type IRC. Ces derniers affirment que le trafic IRC malveillant peut être différencié du trafic légitime. Rishi [GH07] détecte les botnets en analysant les surnoms utilisés par les membres d'un canal IRC. Ils supposent que les surnoms utilisés par les bots présentent une syntaxe différente de ceux des utilisateurs légitimes. Par ailleurs, Livadas et al. [LWLS06] proposent d'utiliser une machine d'apprentissage pour classifier le trafic IRC légitime. Ces approches ne peuvent pas détecter les botnets qui utilisent des canaux IRC chiffrés

D'autres approches ont été proposées pour détecter les canaux IRC de botnet qu'il soit chiffré ou non [SWLL06] [SLWL08] [LRG11]. Dans ces approches, les chercheurs supposent que les informations contenues dans la couche transport (TCP) des canaux IRC (nombre de paquets, leur taille moyenne ...) sont différentes de ceux des canaux légitimes. Ces approches classifient le trafic réseau suivant le type d'applications avant de filtrer le trafic réseau qui se rapproche le plus aux applications légitimes définies, telles que : le transfert de fichier ou la messagerie instantanée. Le trafic réseau restant et non filtré est corrélé afin d'identifier les comportements similaires à un des membres d'un botnet.

Pour détecter les canaux IRC de botnet dans un réseau à large échelle, Karasidis et al. [KRH07] proposent une méthode qui isole les machines qui génèrent du trafic d'attaque comme l'envoi de spam, le scan du réseau, ou le déni de service. Ensuite, ils corrèlent les connexions de ces machines afin de détecter les membres du même botnet et leur serveur de contrôle et commande.

Avec la popularité du protocole HTTP au sein des botnets, plusieurs approches ont été proposées pour détecter les canaux de contrôle basés sur ce protocole. Perdisci et al. [PLF10] [PAG12] proposent de générer des signatures sur le trafic HTTP généré par des logiciels malveillants exécutés dans des environnements contrôlés. Ces signatures sont ensuite utilisées pour détecter de nouveaux botnets engendrant du trafic réseau ressemblant à ces signatures. Par contre, ces méthodes ne peuvent pas détecter le trafic de botnets qui se basent sur un canal de contrôle chiffré parce qu'ils ne peuvent pas générer des signatures sur du trafic chiffré.

Les opérateurs de certains botnets, pour augmenter la résilience de leur réseau et pour distribuer la charge de travail, utilisent plusieurs serveurs de contrôle et commande. Ils distribuent la charge réseau en se basant sur le protocole DNS. Les membres du botnet génèrent régulièrement des requêtes DNS vers le nom de domaine du serveur de contrôle afin de récupérer son adresse. Les systèmes de détection, décrits précédemment, basés sur les flux réseau ne peuvent pas regrouper les connexions vers différents serveurs de contrôles du même botnet. Elles identifient le changement d'adresse IP du serveur distant comme un nouveau flux. Afin de regrouper les flux des membres du même botnet utilisant plusieurs serveurs et ainsi de leur appliquer des algorithmes de classification, Fedynyshyn et al.[FCT11] proposent de regrouper les flux réseau par rapport à leur nom de domaine DNS.

Plusieurs approches ont été proposées pour détecter les botnets en utilisant le trafic DNS. Elles permettent d'identifier les noms de domaine malveillants associés aux serveurs de contrôle de botnets. Exposure [BKKB11] évalue plusieurs propriétés en rapport avec la valeur et la temporalité des champs des réponses DNS. Notos [APD+10] calcule de manière dynamique la réputation d'un nom de domaine en analysant l'historique et la

réputation des différentes adresses IP qui lui ont été associées.

Exposure [BKKB11] évalue plusieurs paramètres d'une réponse DNS afin de qualifier un nom de domaine. Ces paramètres sont :

- basés sur la temporalité : afin d'identifier les noms de domaine qui sont soudainement populaires, et qui peuvent correspondre à des noms de domaine malveillants. Le système calcule le nombre de requêtes reçues vers ce nom de domaine à chaque unité de temps ;
- basés sur les réponses DNS : pour identifier les anomalies dans l'association entre le nom de domaine et les adresses IP telles qu'un nombre élevé d'adresses IP associées, la distribution géographique de ces adresses IP, et le nombre de noms de domaine associés à ces adresses IP ;
- basés sur la durée de validité des réponses (TTL) : pour identifier les noms de domaine qui sont associés à des réponses qui ont une durée de validité faible ou que cette durée change constamment.
- basés sur la syntaxe du nom de domaine : afin d'identifier le pourcentage de mots prononçables et le pourcentage de chiffres contenus dans le nom de domaine.

Cette méthode génère du trafic supplémentaire parce qu'elle émet des requêtes DNS afin d'extraire les noms de domaine associés à certaines adresses IP lors de l'analyse. Elle génère aussi des requêtes vers des moteurs de recherche afin de déterminer la popularité d'un nom de domaine. Elle détermine cette popularité en se basant sur le nombre de résultats retournés.

Ces méthodes de détection sauvegardent pour une certaine durée les informations contenues dans les réponses envoyées aux membres du réseau. Ceci ne permet pas de passer à l'échelle au niveau d'un opérateur réseau comportant plusieurs millions d'utilisateurs actifs.

Les méthodes de détection de noms de domaine malveillants qui se basent sur la réputation des adresses IP ou de leur sous-réseau ne détectent pas les noms de domaine malveillants associés à des adresses IP saines. De plus, ils ne peuvent pas détecter les noms de domaine associés à des serveurs légitimes compromis.

L'utilisation de la valeur des champs de réponse DNS telle que la variation de la durée de validité d'une réponse (champ TTL) pour la détection des noms de domaine malveillants peut être déjouée par les opérateurs de botnets. Pour cela, ils adaptent la valeur des différents champs de la réponse pour éviter de se faire détecter.

La technique fast-flux utilisée pour masquer l'adresse IP d'un serveur malveillant est utilisée par certains opérateurs de botnet pour masquer l'adresse du serveur de contrôle et commande (C&C). Plusieurs méthodes pour détecter exclusivement des noms de domaine de type fast-flux ont été proposées telles que Fluxor [CTD+10] ou celle proposée par Lin et al. [LLC13]. Ces approches se basent sur l'analyse de la valeur des champs de réponses DNS, et plus précisément celle du champ TTL (Time To Live).

Les logiciels malveillants de certains botnets vérifient périodiquement si l'adresse IP de la machine sur laquelle ils sont installés n'a pas été identifiée. Pour cela, ils effectuent des requêtes DNSBL [dns] (DNS Black Lists : liste noire DNS) vers des entités qui gèrent des listes d'adresses IP identifiées comme infectées. Ramachandran et al. [RFD06] proposent de se baser sur ces requêtes DNSBL afin d'identifier les machines infectées.

Choi et al. [CLLK07] proposent une méthode de détection de botnet en identifiant la synchronisation des requêtes de ses membres. Les auteurs supposent que le nombre de machines qui sollicitent un nom de domaine de serveur de C&C est constant dans le temps puisque la taille d'un botnet est fixe. Tandis que, les noms de domaine légitimes sont sollicités par un nombre variable d'utilisateurs dans une période de temps. Dans

cette méthode, ils proposent également d'identifier les botnets lors du basculement du nom de domaine de leur serveur de C&C puisque le nombre de machines qui sollicitent les serveurs d'un botnet est fixe.

L'hypothèse, selon laquelle le nombre de machines qui sollicitent le nom de domaine d'un serveur de C&C est fixe n'est pas toujours vérifiée. Les botnets peuvent utiliser plusieurs serveurs de C&C et laisser leurs membres choisir à quel serveur ils souhaitent se connecter. En conséquence, le nombre de machines qui se connectent à chaque serveur de C&C n'est pas constant.

De plus, les noms de domaine de botnet ne sont pas les seuls à avoir un nombre d'utilisateurs constant. Les applications légitimes ont aussi un nombre constant d'utilisateurs et elles se basent également sur le protocole DNS pour atteindre leur serveur. Les noms de domaine associés à ces applications peuvent générer des faux positifs dans la méthode de détection précédente.

Afin de lutter contre les botnets, certains réseaux bloquent les noms de domaine identifiés malveillants. Ils utilisent des listes noires de noms de domaine de botnets fournis par des tiers tel que l'organisation de sécurité Abuse.ch [abu] qui traque les botnets de type Zeus [BOB+10], SpyEye [SEB12], et Palevo (Mariposa Butterfly) [SBBD10].

Pour déjouer ce blocage, des opérateurs de botnet utilisent la technique domain-flux. Dans cette technique, les membres du botnet envoient des requêtes DNS vers une liste de noms de domaine générés au préalable jusqu'à ce qu'ils se connectent au serveur de C&C.

Yadav et al. [YRRR10] et Haddadi et al. [HKZHH13] proposent la détection de noms de domaine malveillants générés algorithmiquement, qui ne sont pas engendrés par des humains. Ils supposent que les opérateurs de botnet génèrent des noms de domaine imprononçable, contrairement à ceux utilisés pour des applications légitimes. Les noms de domaine malveillants algorithmiquement générés ont une distribution alphanumérique différente d'une utilisation légitime puisque leur but est de générer une liste de noms de domaine qui n'a pas été réservé au préalable. Yadav et al. ont étendu leur approche [YR12] en analysant les réponses de type NXDomain pour accélérer la détection. En revanche, si les opérateurs de botnets ont connaissance des méthodes de détection utilisées, ils peuvent les contourner en adaptant l'algorithme de génération pour qu'il génère des noms de domaine avec des distributions alphanumériques qui ne dépassent pas les seuils établis par ces méthodes.

Pour détecter les botnets de type domain-flux, Jiang N. et al. [JCJ+10] et Antonakakis et al [APN+12] utilisent les réponses DNS de type NXDomain pour identifier les machines infectées. Puisque la majorité des noms de domaine générés ne sont pas réservés, les membres de ces botnets reçoivent un nombre important de réponses de type NXDomain.

L'association entre les noms de domaine inexistants et les utilisateurs peut être représentée dans un graphe biparti, dans lequel l'ensemble des noms de domaine inexistants est un sous-ensemble et les membres du réseau constituent un deuxième sous-ensemble du graphe. Les arêtes représentent la requête d'un utilisateur vers un nom de domaine inexistant. Les méthodes précédentes représentent ce graphe biparti par une matrice.

Par ailleurs, Antonakakis et al [APN+12] ont utilisé les propriétés statistiques pour grouper les noms de domaine du même botnet. Ces derniers ont aussi utilisé les chaines de Markov cachées pour identifier les noms de domaine de serveur de C&C.

Les opérations de corrélation dans les matrices représentants le graphe biparti de l'association entre les membres du réseau et les noms de domaine inexistants demandés, ne passe pas à l'échelle. De plus, les relations entre les membres du réseau et les

noms de domaine inexistants évoluent en permanence ce qui modifie la structure de la matrice. De nouveaux utilisateurs peuvent rejoindre le réseau ou des nouveaux noms de domaine inexistants peuvent être consultés pour la première fois.

Jiang N. et al. [JCJ+10] identifient trois catégories de relation entre les membres d'un réseau et les noms de domaine inexistants :

- Un utilisateur en étoile : un utilisateur qui génère des requêtes DNS vers plusieurs noms de domaine inexistants.
- Un nom de domaine en étoile : représente un nom de domaine sollicité par plusieurs utilisateurs. Ceci correspond à un nom de domaine malveillant bloqué ou à celui d'une application mal-configurée.
- Une communauté : représente une forte interaction entre plusieurs membres du réseau et plusieurs noms de domaine inexistants. Ces membres du réseau envoient des requêtes DNS vers le même ensemble de noms de domaine inexistants. Cette catégorie correspond au comportement des membres d'un botnet de type domain-flux.

3.3 Proposition d'une nouvelle méthode de détection

Un système de détection de botnet au niveau d'un opérateur réseau doit avoir la capacité d'analyser en temps réel le trafic généré par plusieurs millions d'utilisateurs afin d'identifier les membres de botnets. Cette analyse de trafic doit prendre en considération les contraintes liées à la préservation de la vie privée des utilisateurs du réseau par la protection de leur identité ainsi que de leur trafic.

Les membres d'un botnet se connectent à leur serveur de C&C afin de recevoir les dernières commandes. Ceci quelle que soit l'attaque générée par ce botnet. Le trafic réseau de cette connexion ressemble à celui des applications légitimes. En effet, les botnets se basent sur des protocoles usuels tels que HTTP et DNS pour leur communication avec leur serveur de C&C.

La majorité des méthodes de détection de botnets proposées dans l'état de l'art ne peuvent pas analyser en temps réel le trafic généré par un opérateur réseau. De plus, ces méthodes ne prennent pas en compte les contraintes liées à la préservation de vie privée des utilisateurs et de la non-conservation de leur trafic réseau.

L'objectif de nos travaux est l'identification des botnets, c'est-à-dire les machines infectées et leur serveur de contrôle, en se basant sur le trafic d'un réseau à large échelle. Notre méthode doit prendre en compte les contraintes liées à un déploiement d'un système d'analyse du trafic d'un réseau d'opérateur.

Comme la majorité des botnets utilisent le protocole DNS pour atteindre leurs serveurs de contrôle et que le trafic généré par ce protocole ne représente que 1% du trafic internet global, nous proposons dans notre méthode de détection de botnet d'analyser ce protocole. Ce trafic doit être capturé entre les serveurs DNS et les membres du réseau afin de pouvoir analyser les informations contenues dans les requêtes et les réponses.

Le trafic DNS généré par les botnets diffère peu de celui des applications légitimes. Les machines infectées envoient des requêtes au serveur DNS pour résoudre le nom de domaine du serveur de C&C. En conséquence, pour lutter contre les botnets, plusieurs listes noires de noms de domaine malveillants sont apparues. Les entreprises se basent sur ces listes pour bloquer l'accès aux noms de domaine de serveurs malveillants.

Pour contourner ce blocage, une technique appelée domain-flux a été conçue par les opérateurs de botnets. Dans cette technique, les machines infectées génèrent périodi-

quement une liste de noms de domaine. Ensuite, elles envoient des requêtes vers le serveur DNS pour résoudre les noms de domaine de la liste jusqu'à ce qu'ils se connectent au serveur de C&C. Au cours de leur recherche du serveur de contrôle, les machines infectées envoient des requêtes vers des noms de domaine non réservés et reçoivent en conséquence de la part du serveur DNS des réponses spécifiques de type NXDomain.

Nous proposons une méthode de détection de botnet de type domain-flux qui se base dans un premier temps sur les erreurs de résolution DNS. Nous nous basons sur les erreurs de résolution pour trois raisons principales :

- Ils représentent uniquement 15% de la totalité des réponses DNS. Ceci permet de passer à l'échelle d'un grand opérateur réseau de plusieurs millions d'utilisateurs.
- Les erreurs ont une signification moins importante sur les habitudes de connexions des utilisateurs. Ainsi, leur analyse représente un gain pour la préservation de la vie privée des utilisateurs.
- Les erreurs générées par les applications légitimes ont une grande dispersion contrairement aux erreurs générées par les botnets de type domain-flux.

Afin d'éviter la confusion avec les noms de domaine légitimes utilisée comme ceux utilisés par les CDN et pour compliquer les techniques d'évasion, notre méthode de détection ne se base pas sur les propriétés des réponses DNS ou sur la syntaxe des noms de domaine. Elles se basent uniquement sur le comportement des machines infectées.

Notre système de détection doit être déployé idéalement à proximité d'un serveur DNS d'un grand opérateur réseau afin d'analyser le trafic DNS de ses utilisateurs. Elle analyse exclusivement les réponses DNS envoyées depuis les serveurs DNS vers les utilisateurs du réseau pour identifier les membres d'un botnet de type domain-flux ainsi que les noms de domaine malveillants associés.

Notre méthode de détection de botnet se base sur l'analyse des réponses DNS des noms de domaine inexistants dans un premier temps, et sur les réponses DNS des noms de domaine existants dans un deuxième temps. Nous nous basons sur les erreurs de résolution pour identifier les membres d'un même botnet. Ensuite, nous utilisons les réponses DNS réussies pour identifier le serveur de contrôle du botnet.

Notre système de détection est divisé en 4 couches, voir figure 3.2 : la couche d'anonymisation, la couche de filtrage de trafic, la couche de construction de communautés, et la couche de détection de noms de domaine malveillants. La couche d'anonymisation est transversale et traverse toutes les couches. La deuxième couche filtre le trafic réseau inemployé par notre méthode de détection. La troisième couche identifie les machines infectées par des botnets de type domain-flux et groupe les membres d'un même botnet dans une communauté. La dernière couche identifie les noms de domaine de contrôle de botnets associés aux communautés construites.

La première couche est transverse à tout le système. Elle s'assure de la confidentialité et de l'anonymisation du trafic utilisé sur tout le processus de détection : (1) elle anonymise l'adresse IP des machines avant de les ajouter dans le système ; (2) et représente les noms de domaine sollicités par chaque machine dans des structures de données probabilistes dans les deux dernières couches. La structure de donnée probabiliste permet de préserver la confidentialité des données des utilisateurs.

La deuxième couche filtre le trafic qui n'est pas utilisé par notre système de détection. Elle filtre différemment les deux types de réponses DNS. Elle exclut de l'analyse les noms de domaine inexistants mal-formés. Elle exclut aussi de l'analyse les noms de domaine sollicités par les machines non soupçonnées d'être infectées. Le filtrage à ce niveau de l'analyse permet d'optimiser les performances de notre système.

La troisième couche de notre système détecte les machines qui génèrent un tra-

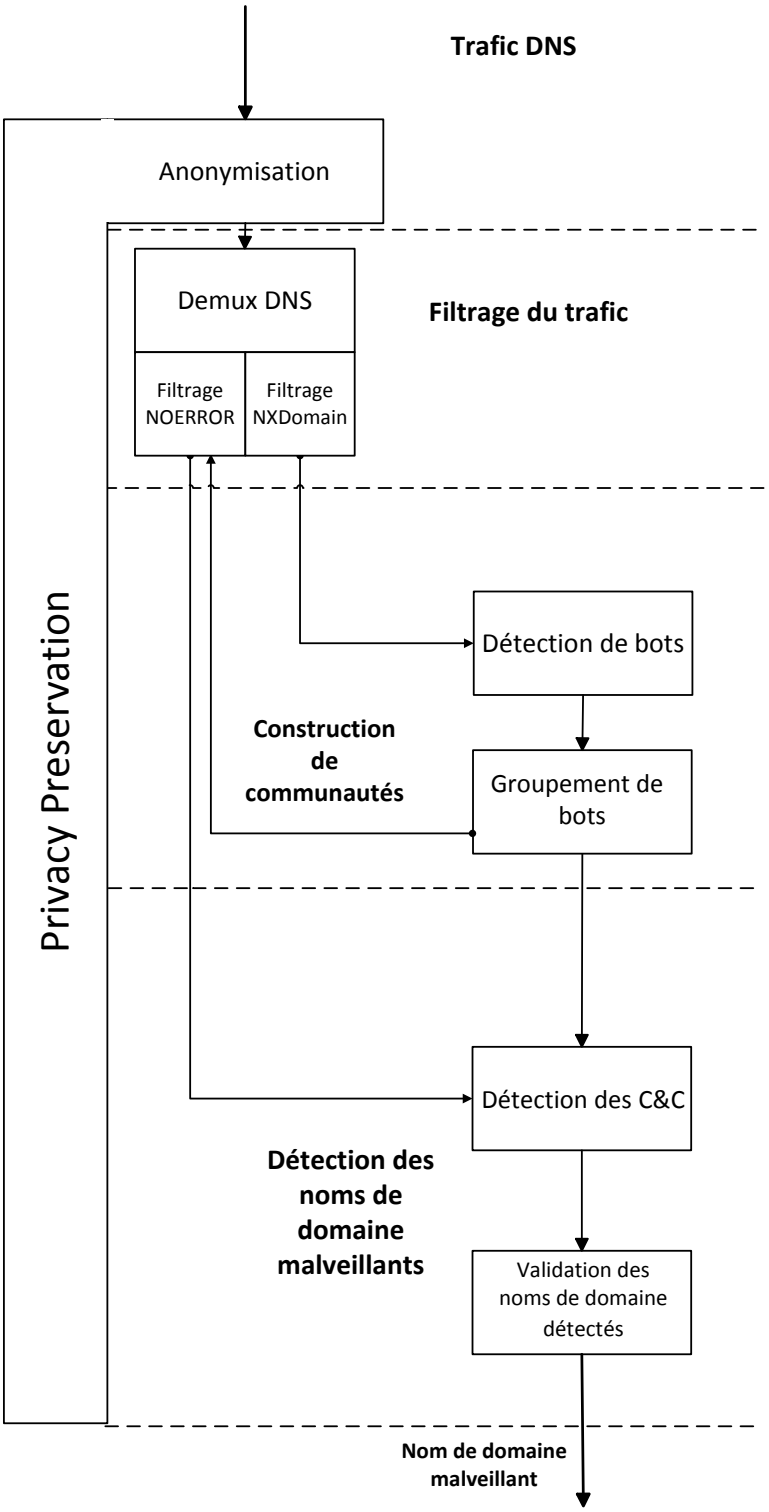

FIGURE 3.2 – Architecture en couches de notre système de détection

fic réseau équivalent à ceux d'un botnet de type domain-flux. Elle groupe ensuite les membres d'un même botnet en communautés en comparant leur trafic réseau généré. Cette couche reçoit les informations sur les réponses DNS de noms de domaine inexistants de la part de la couche de filtrage de trafic. Les machines qui génèrent un nombre élevé de requêtes vers des noms de domaine inexistants sont déclarées suspects. Les listes des noms de domaine inexistants sollicités par chaque machine suspecte sont comparées pour construire des communautés contenants les membres d'un même botnet. La communauté construite est ensuite envoyée vers la couche de détection de noms de domaine malveillants et à la couche de filtrage de trafic.

La dernière couche identifie les noms de domaine malveillants associés aux communautés construites. Elle reçoit les informations sur les réponses DNS des noms de domaine existants envoyés vers des membres de communautés identifiées. En se basant sur ces informations, elle identifie les noms de domaine qui sont sollicités par la majorité des membres d'une communauté. Ces noms de domaine correspondent soit à des noms de domaine populaires c'est-à-dire qu'ils ont été sollicités lors d'une utilisation légitime, ou à des noms de domaine malveillants de botnets qui ont été sollicités par des logiciels malveillants. Elle différencie les deux types de noms de domaine de façon dynamique.

3.3.1 Privacy Preservation

La mise en oeuvre de méthodes d'anonymisation du trafic des machines est une condition nécessaire pour l'implémentation d'un système au niveau d'un opérateur réseau pour analyser de trafic de tiers. Cette condition est liée à la fois à des contraintes morales (contrat de confiance entre l'opérateur et ses abonnés) et à des contraintes règlementaires. C'est ainsi le cas en France où l'article L34-1 du "Code des postes et des communications électroniques" impose que les opérateurs de communications électroniques "effacent ou rendent anonyme toute donnée relative au trafic". Afin de répondre à ces contraintes, nous définissons une couche, que l'on appellera "Privacy Preservation", et qui s'étend de manière transversale à l'ensemble de notre système de manière à rendre confidentielle l'adresse IP des machines et pour éviter le stockage en clair de leur trafic réseau.

Les fonctions apportées par cette couche seront ainsi les suivantes : elle anonymise les adresses IP des machines lors de la capture du trafic, et s'assure que les informations récupérées lors de l'analyse sont stockées dans des structures de données probabilistes dans les deux dernières couches de notre système (construction de communauté, et détection de noms de domaine malveillants).

Anonymisation des adresses IP

Notre méthode de détection se base exclusivement sur l'identification de l'association entre les membres d'un botnet de type domain-flux et de leurs noms de domaine malveillants. Ainsi, pour identifier cette association, on compare périodiquement les noms de domaine générés par chaque machine caractérisée par son adresse IP.

Pour détecter les noms de domaine malveillants, notre système de détection n'a pas besoin de connaitre l'adresse IP réelle des machines. Il a besoin seulement qu'un identifiant soit associé à chaque machine tout au long du processus de détection pour pouvoir les différencier. L'identifiant associé à chaque machine est le résultat de la fonction de hachage de le concaténation de son adresse avec un sel.

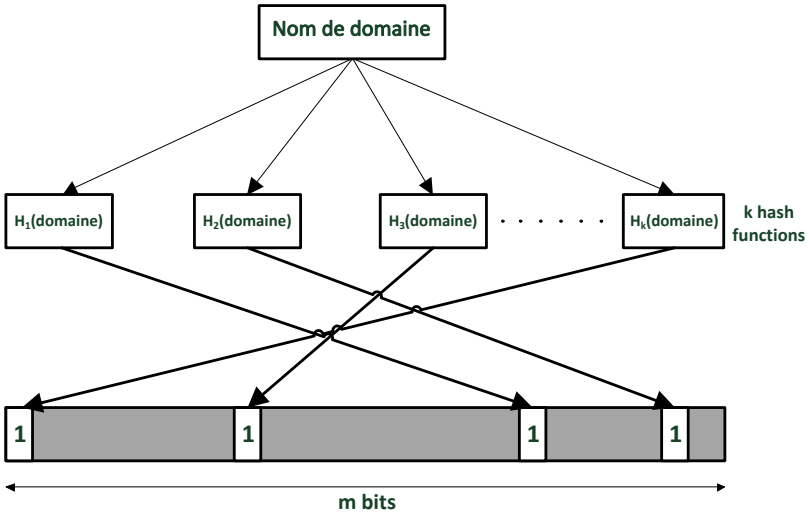

FIGURE 3.3 – Insertion d'un nom de domaine dans un filtre de Bloom binaire de taille m avec k fonctions de hachages

Le sel utilisé dans notre méthode est une valeur aléatoire calculée périodiquement pour éviter que les attaquants puissent remonter à l'adresse IP des machines. Quand cette valeur est recalculée, tous les identifiants du système deviennent obsolètes, et sont remis à zéro. Les informations relatives aux machines sont aussi supprimées.

L'anonymisation des adresses IP est nécessaire dans le cas où la capture du trafic n'est pas accomplie dans le même processus que l'analyse. Dans ce cas de figure, le changement de sel est communiqué au système de détection afin qu'ils puissent se mettre à jour. Si la capture du trafic a lieu dans le même processus que toutes les couches de notre système de détection, cette étape peut être évitée puisque le processus reçoit déjà les adresses IP des machines.

Utilisation des filtres de Bloom

Afin de protéger la vie privée des utilisateurs des machines et pour optimiser la mémoire de notre système de détection, on utilise des filtres de Bloom binaires [Blo70], structures de données probabilistes qui génèrent un faible taux de faux positifs, pour représenter les noms de domaine sollicité par les utilisateurs du réseau. Ces structures de données permettent de représenter les éléments d'un ensemble dans un espace mémoire de taille fixe sans générer de faux négatifs.

Un filtre de Bloom binaire est un vecteur binaire de m bits associés à k fonctions de hachages [BM04] comme illustré dans la figure 3.3. Le vecteur est initialisé et tous ses champs sont mis à 0. Lors de l'insertion d'un élément dans un filtre de Bloom, on applique les différentes fonctions de hachage sur l'élément. Les valeurs retournées par ses fonctions sont des entiers qui appartiennent à l'intervalle [0, m-1]. Les champs du vecteur qui correspondent à ces valeurs sont mis à 1.

Les filtres de Bloom binaires n'admettent que deux opérations : l'ajout et le test d'appartenance d'un élément. La suppression d'un élément n'est pas possible dans un filtre

de Bloom binaire. Il ne suffit pas de remettre les valeurs des champs correspondants à l'élément dans le vecteur à 0 parce qu'un champ peut avoir été initialisé par plusieurs éléments de l'ensemble.

Comme la suppression d'éléments n'est pas possible dans les filtres de Bloom binaires, on remet à zéro périodiquement le vecteur du filtre de Bloom pour rafraîchir le trafic réseau représenté. Par conséquent, notre méthode de détection ne prend en compte que les dernières réponses des machines lors de l'analyse.

Les filtres de Bloom ne génèrent pas de faux négatif, c'est-à-dire que le test d'appartenance d'un élément d'un ensemble représenté par le filtre de Bloom sera toujours positif. Par contre, il engendre des faux positifs avec une certaine probabilité p. Un test d'appartenance positif d'un élément signifie qu'il appartient à l'ensemble représenté par le filtre de Bloom avec une probabilité (1-p).

La probabilité que la valeur d'un champ du vecteur soit 0 après l'insertion de n éléments dans un filtre de Bloom avec k fonctions de hachages et un vecteur binaire de taille m est :

$$\left(1 - \frac{1}{m}\right)^{kn}$$

Ainsi, la probabilité que la valeur d'un champ soit 1 est de :

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)$$

La probabilité d'un faux positif est alors :

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k}$$

Il est très difficile d'inverser un filtre de Bloom binaire pour retrouver les éléments de l'ensemble représenté. Particulièrement, lorsque la taille de l'ensemble est largement supérieure à la taille du vecteur. Dans notre méthode, les filtres de Bloom représentent les noms de domaine demandés par les machines du réseau. Le nombre de ces noms de domaine dépasse le million et la taille du vecteur utilisé est l'ordre d'un millier. En conséquence, chaque champ du vecteur peut correspondre à plusieurs milliers de noms de domaine. Cette propriété permet de garantir la confidentialité du trafic généré par les machines dans notre système puisque l'analyse d'un vecteur du filtre de Bloom ne permet pas de déchiffrer les noms de domaine demandés.

Les filtres de Bloom sont la représentation des éléments d'un ensemble dans un vecteur à taille fixe. Le remplissage du vecteur est effectué au moyen de fonctions déterministes. Ainsi si deux ensembles sont similaires, leurs filtres de Bloom utilisant les mêmes paramètres (taille du vecteur, nombre et type de fonctions de hachage) seront similaires. Deux ensembles légèrement différents seront représentés avec des filtres de Bloom légèrement différents. Par contre, deux filtres de bloom similaires représentent les mêmes ensembles d'éléments avec une certaine probabilité.

Afin d'accélérer la procédure de corrélation lors de la construction de communauté, nous comparons entre les vecteurs de filtres de Bloom associés aux machines au lieu de comparer leur trafic réseau généré.

Afin de représenter uniquement le trafic réseau récent des machines et pour réduire le nombre de faux positifs dus à la taille des vecteurs, les filtres de Bloom sont remis à zéro périodiquement. Ils sont aussi remis à zéro lorsque le sel de la fonction de hachage utilisé dans le module d'anonymisation des adresses IP est renouvelé.

3.3.2 Filtrage du trafic

Cette couche reçoit les réponses DNS capturées dans le réseau, et permet de les filtrer de manière à ne conserver que les informations qui seront utiles pour la suite de l'analyse. Comme détaillé ci-après, le type de filtrage variera suivant le type de réponse considérée. Ainsi, les réponses de type NXDomain seront filtrées sur la base de la validité du nom de domaine, et les réponses de type NOError seront filtrées sur la base de l'identifiant de la machine destinataire de la réponse.

À la suite du filtrage, chaque type de réponse est envoyé à la couche correspondante : les réponses NXDomain sont envoyées à la couche de construction de communautés et les réponses NOError sont envoyées vers la couche de détection de noms de domaine malveillants.

Filtrage NxDomain

Les membres d'un botnet de type domain-flux envoient des requêtes DNS vers la liste des noms de domaine générés de manière algorithmique afin d'atteindre leur serveur de contrôle et commande. L'opérateur du botnet ne réserve qu'un nom de domaine afin de l'associer au serveur de C&C.

Les noms de domaine générés doivent être syntaxiquement corrects afin de pouvoir être réservé par l'opérateur du botnet. En effet, un nom de domaine mal-formé ne peut pas être réservé au niveau DNS pour l'associer au serveur de C&C. On exclut de notre analyse les noms de domaine mal-formés puisqu'ils ne sont pas reliés à des botnets de type domain-flux.

Un nom de domaine syntaxiquement correct :

- Contient un Top Level Domain (TLD) valide défini par l'ICANN [ICA]. Le TLD est le dernier élément de l'arborescence DNS avant la racine. Par exemple "com" et "org" sont des TLD valides.
- Contient au minimum un Second Level Domain (2LD). Dans le nom de domaine "exemple.fr", "exemple" est le 2LD et "fr" est le TLD.
- Ne contient pas de caractère de contrôle.

On résume le résultat de notre filtrage des noms de domaine inexistants sur du trafic DNS d'un opérateur réseau à large échelle dans le tableau 3.1. Les noms de domaine avec des TLD invalides représentent 23% des noms de domaine inexistants, ceux qui n'ont pas de TLD en représentent 10%, et ceux contenants des caractères de contrôle représentent 1%. Les noms de domaine restants (63%) sont syntaxiquement corrects.

TABLE 3.1 – Filtrage NXDomain

caractères invalides	1%
TLD absent	10%
TLD non valide	23%
FQDN valide	66%

Filtrage NoError

Les membres d'un botnet de type domain-flux reçoivent les réponses de type NOError après les réponses NXDomain. En effet, après la localisation du serveur de C&C, les membres du botnet n'ont plus besoin de générer des requêtes DNS. En conséquence,

notre méthode de détection analyse uniquement les réponses NOError des machines suspectées d'être infectées et membres de communautés construites et exclut de l'analyse les réponses des autres machines.

L'analyse des réponses NOError uniquement des machines infectées permet de :

- garantir la vie privée des utilisateurs.
- analyser le trafic réseau d'un large opérateur puisque les erreurs ne représentent que 15% des réponses DNS.
- utiliser des structures de données de taille plus importantes que celles utilisées pour représenter les noms de domaine inexistants.

La liste des machines infectées est reçue de la part de la couche de construction de communauté.

3.3.3 Construction de communautés

Cette couche reçoit les réponses de type NxDomain contenants les noms de domaine syntaxiquement valides, et l'identifiant de la machine destinataire de la réponse. Elle construit les communautés de bots en deux phases : elle identifie les utilisateurs potentiellement infectés, puis elle compare leur trafic réseau généré afin d'identifier les membres du même botnet.

Lors de la réception d'une réponse NxDomain, nous ajoutons le nom de domaine dans le filtre de Bloom associé à la machine destinataire de la réponse. Ensuite, on analyse ce filtre de Bloom afin de vérifier si le trafic réseau de la machine correspond à un comportement de membres d'un botnet de type domain-flux. Les machines potentiellement infectées sont analysées périodiquement afin de construire les communautés de bots.

Détection de bots

Nous créons et associons à tout les membres actifs du réseau, un filtre de Bloom afin de représenter l'ensemble des noms de domaine inexistants demandés et syntaxiquement valide. Les filtres de Bloom associés aux membres du réseau sont de même taille pour pouvoir les comparer ultérieurement.

Afin de pouvoir représenter l'ensemble des utilisateurs actifs du réseau dans la mémoire du système, nous créons des filtres de Bloom d'un millier de bits avec une seule fonction de hachage. Lors de notre analyse du réseau, nous avons remarqué que cette taille est suffisante pour représenter les noms de domaine inexistants générés par les machines saines. De plus, cette taille de vecteur permet de représenter les noms de domaine de membres du botnet de type domain-flux Conficker.C, car ils génèrent en moyennes 500 requêtes DNS différentes [LW09] par jour.

Puisque la suppression d'éléments n'est pas possible dans les filtres de Bloom, nous les remettons à zéro régulièrement pour qu'ils représentent uniquement les derniers noms de domaine inexistants sollicités par les machines du réseau. Par ailleurs, nous supprimons les informations associées aux machines inactives afin d'optimiser l'espace mémoire du système de détection.

Lors de la réception d'une réponse de type NXDomain, nous vérifions que l'identifiant correspondant à la machine est présent dans le système et qu'un filtre de Bloom lui est associé. S'il n'est pas présent, nous le créons et lui associons un filtre de Bloom initialisé. Nous ajoutons le nom de domaine inexistant dans le filtre de Bloom en initialisant à 1 le résultat de la fonction de hachage pour ce nom de domaine, quelle que soit son ancienne

valeur. Par conséquent, un nom de domaine sollicité plusieurs fois par la même machine sera représenté une seule fois dans le filtre de Bloom.

Pour évaluer l'infection d'une machine, nous définissons le taux de remplissage du filtre de Bloom τ au nombre de bits à 1 sur le vecteur divisé par la taille de ce vecteur. Ceci correspond dans ce cas à la quantité de noms de domaine inexistants sollicités récemment par la machine considérée.

Après l'ajout du nom de domaine dans le filtre de Bloom, nous évaluons son taux de remplissage afin d'identifier les utilisateurs ayant un comportement correspondant à celui des membres de botnet de type domain-flux.

Pour détecter les machines susceptibles d'être membre d'un botnet de type domain-flux, nous définissons un seuil d'infection α. Si le taux de remplissage du filtre de Bloom de la machine dépasse ce seuil, son identifiant est rajouté à la liste des machines suspectées d'être infectées. Nous avons estimé empiriquement la valeur de α à 0.5%, qui correspond à 5 requêtes vers des noms de domaine inexistants.

Problématique des proxys Notre système de détection est idéalement localisé près du serveur DNS d'un opérateur réseau. Il assigne à chaque adresse IP un identifiant unique afin de détecter les machines infectées par un botnet de type domain-flux. Ces adresses IP publiques peuvent être partagées par plusieurs machines qui auront le même identifiant dans notre système, que nous appelons proxys. Par conséquent, les filtres de Bloom associés à un proxy représentent les noms de domaine inexistants sollicités par plusieurs machines.

Nous avons remarqué lors de notre analyse que les proxys génèrent un nombre important de requêtes vers des noms de domaine inexistants dans une courte période de temps. En effet, leurs filtres de Bloom associés sont remplis rapidement, c'est-à-dire que la majorité des champs du vecteur sont à 1. Par conséquent, les filtres de Bloom associés à des proxys sont similaires, et génèrent des communautés contenants exclusivement des proxys, au lieu de contenir les membres d'un même botnet.

Puisque le nombre de machines derrière chaque membre de cette communauté de proxy est élevé, les noms de domaine existants qu'ils sollicitent convergeront sur plusieurs points correspondants à des noms de domaine légitimes. Ainsi, le non-filtrage des proxys générera des faux positifs.

Pour filtrer les proxys, nous définissons un seuil β. Si le taux de remplissage d'un filtre de Bloom dépasse ce seuil, on écarte l'identifiant correspondant de notre analyse, et il est retiré de la liste des machines suspectes, s'il est dedans. Nous avons estimé de manière empirique la valeur de β à 70%.

En résumé, on déclare une machine suspecte si son taux de remplissage τ est compris entre α et β.

$\alpha \leq \tau \leq \beta$.

Groupement de bots

Les membres d'un même botnet de type domain-flux génèrent périodiquement des listes de noms de domaine inexistants similaires. Ils envoient continuellement des requêtes DNS vers ces noms de domaine jusqu'à ce qu'ils atteignent leur serveur de C&C. En conséquence, ils reçoivent des réponses de type NXDomain de noms de domaine inexistants similaires avant de recevoir une réponse de type NOError du nom de domaine du serveur de C&C.

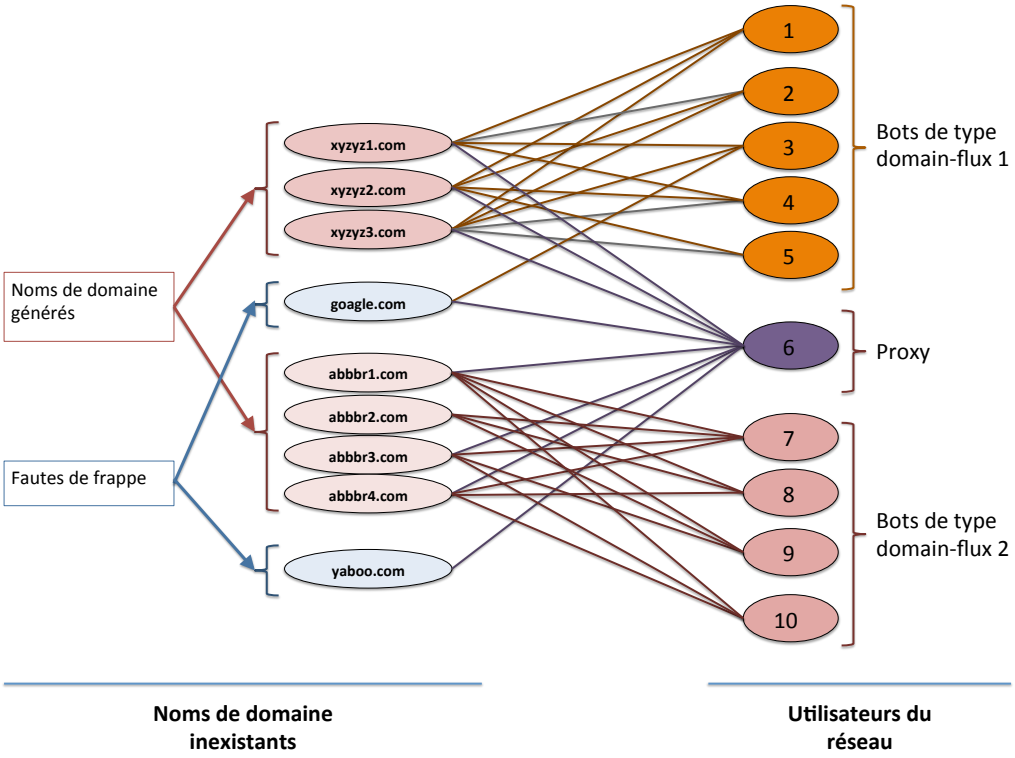

FIGURE 3.4 – Association entre les membres de botnets de type domain-flux et les noms de domaine inexistants

Nous analysons périodiquement le trafic réseau généré par les machines suspectées d'appartenir à un botnet reçu par le module précédent. L'objectif de l'analyse est l'identification des groupes de machines appartenant au même botnet de type domain-flux.

Dans un botnet de type domain-flux, les machines infectées et les noms de domaine inexistants qu'ils sollicitent peuvent être représentés à l'aide d'un graphe biparti, voir figure 3.4. Dans notre système de détection, nous pouvons représenter les identifiants des machines dans un sous-ensemble et les noms de domaine inexistants demandés dans le deuxième sous-ensemble du graphe.

La construction de communautés correspond dans la figure 3.4 à la recherche de sous-graphes, où un groupe de machines est relié au même ensemble de noms de domaine inexistants, tout en excluants les proxys. Ces sous-graphes représentent la relation entre les membres d'un botnet de type domain-flux et les noms de domaine inexistants générés.

Le filtre de Bloom associé à une machine représente les noms de domaine sollicités par cette dernière. Ainsi, pour comparer l'ensemble des noms de domaine sollicités par les machines du réseau, il suffit de comparer leurs filtres de Bloom associés puisque nous avons utilisé les mêmes paramètres (taille de vecteur, nombre et type de fonctions de hachage) pour les créer.

Nous filtrons implicitement les machines qui génèrent des requêtes vers des noms de domaine inexistants légitimes. Autrement dit, des noms de domaine qui ne sont pas générés par des logiciels malveillants, par exemple : les fautes de frappe lors de l'écriture du nom de domaine dans le navigateur. Ces derniers ne peuvent pas aboutir à la construction d'une communauté dans notre système puisqu'il est peu probable que plu-

sieurs utilisateurs fassent les mêmes fautes de frappe dans le même intervalle de temps réduit.

Les membres d'un même botnet de type domain-flux ne génèrent pas tous des requêtes vers des ensembles de noms de domaine identiques. Ces ensembles diffèrent sensiblement pour différentes raisons :

- Une absence de synchronisation des algorithmes de génération de noms de domaine (DGA) des machines infectées.
- Des noms de domaine inexistants supplémentaires peuvent être générés par l'utilisateur de la machine de manière légitime qui s'ajoute à ceux du logiciel malveillant.
- L'ordre des noms de domaine lors de la génération peut être différent chez les logiciels malveillants. Par conséquent, certaines machines atteignent le serveur de C&C avant les autres. En effet, une machine peut générer 10 requêtes DNS vers des noms de domaine inexistants alors qu'une autre ne génère que 6 requêtes.
- Certains botnets comme Conficker.C [LW09] génèrent un nombre important de noms de domaine, mais chaque bot envoie des requêtes de résolution DNS vers une sous-partie de l'ensemble généré, défini par une deuxième fonction aléatoire spécifique à la machine infectée. Par conséquent, les membres de ce botnet n'envoient pas des requêtes vers le même ensemble de noms de domaine inexistants.

Afin d'identifier les filtres de Bloom similaires, nous avons défini un taux de similarité. Ce taux de similarité est dérivé de la distance de Hamming qui est largement utilisé dans les correcteurs de code. Nous calculons le taux de similarité entre deux vecteurs binaires de même taille. La valeur de ce taux est comprise entre 0 et 1. S'il est proche de 0, cela veut dire que les deux vecteurs sont différents. Par contre, s'il est proche de 1, alors les deux vecteurs sont similaires.

La distance de Hamming est calculée entre deux vecteurs binaires de même taille. Il correspond au nombre de bits différents entre deux vecteurs. Il correspond aussi au poids de Hamming du vecteur résultant du "OU exclusif" XOR des deux vecteurs. Le poids de Hamming d'un vecteur correspond au nombre de bits positionnés à 1.

Poids de Hamming :

$$weight(10110) = 3$$
$$weight(11111) = 5$$

Distance de Hamming :

$$d(11111, 10110) = 2$$
$$d(01010, 10100) = 4$$

Avant de calculer le taux de similarité, nous calculons le facteur de similarité. Ce facteur de similarité correspond au poids de Hamming appliqué au résultat du ET binaire entre deux vecteurs. Ce taux est le nombre de bits positionnés à 1 dans les deux vecteurs.

facteur de similarité :

$$S(11111, 10110) = 3$$
$$S(01010, 10100) = 0$$

Le taux de similarité correspond à la division du facteur de similarité par le maximum des poids de Hamming des deux vecteurs. Ainsi, le taux de similarité entre les deux vecteurs A et B de même taille est :

$$similarityrate(A, B) = \frac{S(A,B)}{max(weight(A),weight(B))}$$

Afin d'identifier et regrouper les machines appartenant aux mêmes botnets de type domain-flux, nous comparons deux à deux les filtres de Bloom associés aux machines

suspectées d'être infectées. Si le taux de ressemblance dépasse un seuil γ, nous déclarons les deux filtres de Bloom semblables et nous ajoutons leurs identifiants associés au même groupe. Si la taille de ce groupe de machines associé à des filtres de Bloom semblables dépasse le seuil λ, nous déclarons que les membres du groupe appartiennent au même botnet de type domain-flux.

Pour optimiser le processus de corrélation, nous calculons le facteur de similarité entre deux vecteurs seulement s'ils sont susceptibles d'être similaires. Ainsi, avant de calculer le taux de similarité, nous comparons leur taux de remplissage. Si le taux de remplissage du filtre de Bloom le plus élevé divisé par le taux de remplissage du deuxième filtre de Bloom est inférieur à γ, alors le taux de similarité ne peut pas être supérieur à γ.

L'algorithme de création de communautés est détaillé dans algorithm 3. Nous comparons les filtres de Bloom des machines suspectées d'appartenir à un botnet de type domain-flux.

Pour chaque filtre de Bloom x représenté dans le système, nous créons une nouvelle liste Y afin d'y mettre les filtres de Bloom similaires. Nous comparons tous les filtres représentés dans le système avec x. S'ils sont similaires, on les rajoute dans la liste Y. Nous vérifions la similarité des filtres en comparant leur taux de remplissage. Ensuite, nous calculons le taux de similarité pour vérifier s'il dépasse γ.

Après avoir comparé tous les filtres avec x, nous vérifions le nombre d'éléments dans la liste Y construite. Si ce nombre dépasse λ, alors nous déclarons cette liste une communauté de machines infectées par un même botnet de type domain-flux. Ensuite, nous envoyons cette communauté vers la couche de détection de C&C, et vers la couche de filtrage de trafic afin de servir au filtrage des réponses NOError. Pour ne pas avoir de machine appartenant à plusieurs communautés, nous supprimons les membres de la communauté créés ici de la liste des machines suspectées d'être infectées.

Les membres de communautés construites ainsi que le moment de leur détection sont conservés dans la mémoire, afin qu'un même membre n'apparaisse pas dans des communautés différentes. Ceci aurait en effet pour conséquence d'induire en erreur le module de détection de serveur de C&C qui a besoin d'une correspondance entre l'identifiant d'une machine infectée et une communauté.

3.3.4 Détection des noms de domaine malveillants

Après la détection des membres d'un même botnet, nous cherchons à identifier leurs serveurs malveillants associés. Contrairement à la couche précédente, cette couche se base uniquement sur les résolutions DNS réussies, c'est-à-dire sur les réponses de type NOError associées à des membres de communautés détectées.

D'un point de vue statistique, les noms de domaine existants sollicités par les membres d'un botnet convergent soit vers leur serveur de C&C, soit vers des noms de domaine populaires.

Afin d'accélérer le processus d'identification des serveurs malveillants et de garantir la préservation de la vie privée des utilisateurs des machines non infectées par ce type de logiciel malveillant, cette couche analyse seulement les résolutions DNS réussies des communautés construites. Les résolutions DNS réussies des machines non infectées sont exclut de l'analyse dans la couche de filtrage du trafic.

Nous détectons les noms de domaine malveillants en deux étapes : dans une première étape, nous identifions les noms de domaine sollicités par la majorité des membres d'une communauté. Dans la deuxième étape, nous cherchons à distinguer les noms de domaine populaires des noms de domaine malveillants.

Algorithm 3 Algorithme de construction de communautés

γ : le seuil inférieur du taux de similarité.
λ : la taille minimum d'une communauté.
x, y, z : des filtres de Bloom
f : fonction de similarité
T : Listes des filtres des machines suspectes
for x in T **do**
    M : Listes de filtres ressemblants à x
    ajouter x dans M
    for int i < $T.length$ **do**
        $y \leftarrow T[i]$
        if $x \neq y$ **then** ▷ ne pas comparer x avec lui même
            if (($x.weight \geq y.weight$ AND $\frac{x.weight}{y.weight} \geq \gamma$) OR ($y.weight \geq x.weight$ AND $\frac{y.weight}{x.weight} \geq \gamma$)) **then**
                if $f(x, y) \geq \gamma$ **then**
                    ajouter y dans M
                end if
            end if
        end if
    end for
    if $M.length \geq \lambda$ **then**
        M contient une communauté de machines infectées
        for z in M **do** ▷ Effacer les membres de M de la liste T
            Supprimer z de T
        end for
    end if
end for

Étape 1 : Détection de C&C

Nous associons à chaque membre d'une communauté d'utilisateurs un nouveau filtre de Bloom afin de représenter les noms de domaine de résolutions DNS réussies. La taille de ce filtre de Bloom est supérieure à celui utilisé pour représenter les noms de domaine inexistants. Ceci pour deux raisons principales :

- le nombre de machines infectées est moins important que l'ensemble des membres du réseau ;
- nous avons observé lors de notre analyse du trafic DNS que le nombre de noms de domaine existants sollicités par une machine est en moyenne plus important que le nombre de noms de domaine inexistants.

À chaque communauté construite, nous associons également un filtre de Bloom représentant l'ensemble des noms de domaine existants sollicité par ses membres. La taille de ce filtre de Bloom est largement supérieure aux membres de la communauté.

Pour chaque résolution DNS aboutie, nous ajoutons le nom de domaine sollicité au filtre de Bloom associé au destinataire de la réponse, ainsi qu'au filtre associé à la communauté à laquelle appartient le destinataire.

Afin d'éviter de générer les faux positifs, nous excluons de l'analyse les filtres de bloom qui présentent un taux de remplissage élevé. Ces derniers peuvent générer des tests d'appartenance positifs pour des noms de domaine non sollicités par la machine.

Afin de ne pas répéter le processus de détection et pour optimiser les performances du système, on vérifie que le nom de domaine n'a pas été caractérisé auparavant, c'est-à-dire qu'il n'a pas été identifié comme malveillant ou populaire.

Après l'ajout du nom de domaine dans le filtre de Bloom, on vérifie s'il a été demandé par d'autres membres de la communauté de l'utilisateur associés. Pour cela, nous effectuons un test d'appartenance du nom de domaine sur tous les filtres des membres de la communauté. Si l'on constate qu'il a été demandé par la majorité des membres de la communauté, cela indique que le nom de domaine est soit populaire, soit malveillant.

Étape 2 : Validation de nom de domaine détectés

Les noms de domaine résultants de la convergence du trafic des membres d'une communauté correspondent à l'une des catégories suivantes :

1. serveurs de contrôle et commande (C&C) d'un botnet ;
2. serveurs légitimes populaires demandés par un botnet ;
3. serveurs populaires demandés par un utilisateur de manière légitime.

Certains botnets génèrent des requêtes vers des sites web légitimes afin de tester la connexion réseau de la victime, de synchroniser l'horloge de leurs différents membres, ou pour perturber [JHKH11] les systèmes d'analyse ou de détection de botnets.

Lors de notre analyse, nous avons remarqué que les noms de domaine populaires sont sollicités par au moins un membre de toutes les communautés représentées dans notre système, voir figure 3.5. En revanche, les noms de domaine de serveurs de C&C ne sont demandés que par les communautés qui représentent ces botnets.

Nous définissons un indicateur de dispersion Δ pour différentier les noms de domaine populaires de ceux de serveur de C&C. Cet indicateur correspond au taux de communautés contenant un membre ayant sollicité le nom de domaine. Pour calculer Δ, on effectue un test d'appartenance sur les filtres de Bloom associés à chacune des communautés du système.

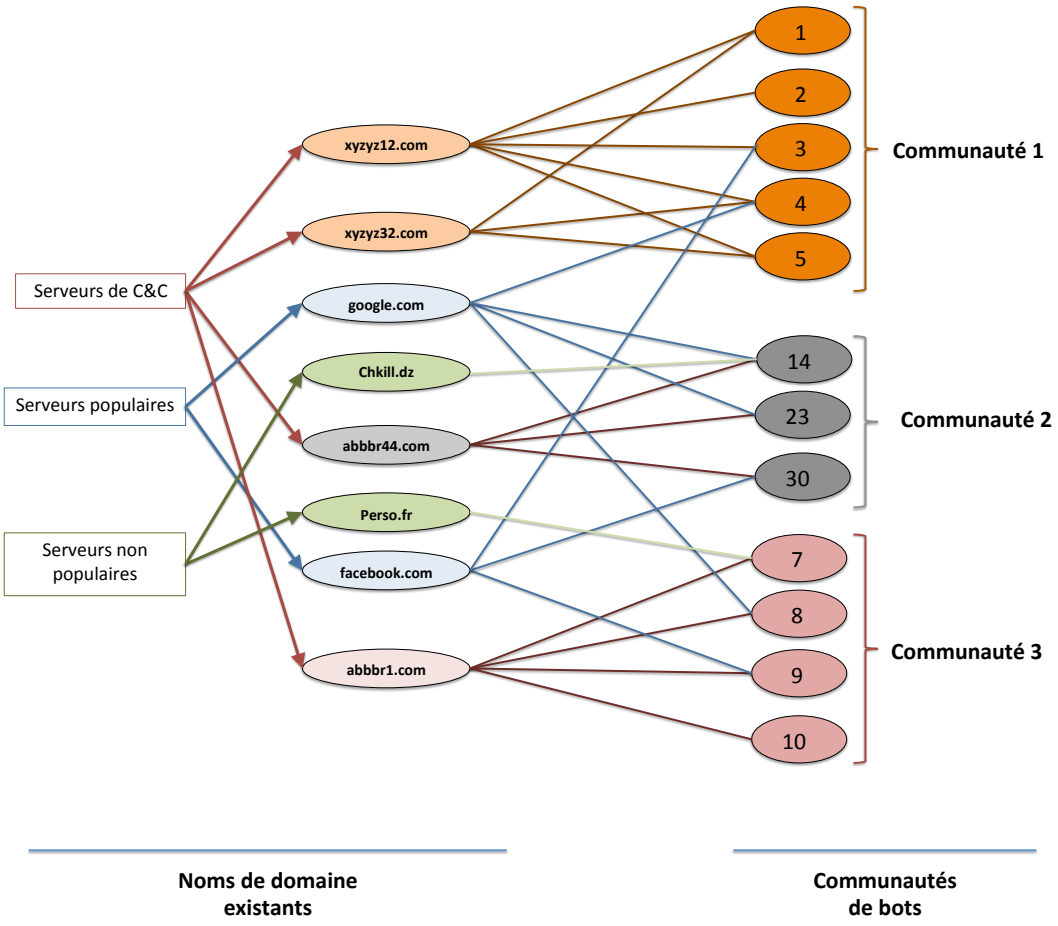

FIGURE 3.5 – convergences des noms de domaine de résolutions DNS réussies de communautés de bots

Si la valeur de $\Delta(domaine)$ est faible, on considère le nom de domaine malveillant puisqu'il a été demandé que par une minorité de communautés. En revanche, si la valeur de $\Delta(domaine)$ est élevé, on considère le nom de domaine populaire et légitime puisqu'il a été demandé par au moins un membre de la majorité des communautés du système.

Si la valeur de l'indicateur de dispersion d'un nom de domaine correspond à une activité légitime, on l'ajoute à une liste blanche dynamique qui contient les noms de domaine caractérisés populaires par notre système. Par ailleurs, les noms de domaine malveillants sont ajoutés à une liste noire pour accélérer le processus d'analyse.

3.4 Évaluation et validation

Pour évaluer notre approche de détection, nous avons analysé deux traces réseaux de 12 et de 1 heure représentant le trafic réseau capturé à proximité d'un serveur DNS d'un opérateur réseau à large échelle contenant plusieurs millions de machines. La première capture date de 2009 et contient plus de 2.5 milliards de paquets DNS avec un débit de plus de 60000 paquets DNS par secondes. La deuxième capture date de 2010 et contient plus de 645 millions de paquets DNS avec un débit de 180000 paquets DNS par secondes.

Les statistiques des captures sont résumées dans le tableau 3.2. Les réponses NXDomain et les réponses NOERROR représentent respectivement 14% et 80% de la totalité des réponses dans les deux captures. Dans ces traces, le serveur DNS est sollicité en permanence par plus de 5 millions de machines différentes.

TABLE 3.2 – Statistiques de capture

	Capture 1	Capture 2
Nombre de paquets	2554952257	645629853
requêtes DNS	1235008617	311332223
réponses NOERROR	995720974	257984280
réponses NXDOMAIN	178995004	44213222

Pour mesurer l'efficacité de notre méthode de détection, nous avons calculé les valeurs des différents paramètres de notre système afin d'optimiser le nombre de noms de domaine malveillants détectés tout en minimisant le nombre de faux positifs. Puisque les performances d'analyse de notre système sont importantes pour un déploiement dans un opérateur à large échelle, nous avons aussi mesuré dans nos analyses la vitesse d'analyse de notre système.

Afin de caractériser les noms de domaine détectés par notre système, nous avons vérifié leur présence dans la liste de noms de domaine populaires fournis par Alexa [ale]. Ceci permet de caractériser les faux positifs puisqu'un site très populaire est considéré comme bien maintenu et fournissant un service légitime non relié à une activité malveillante. Nous avons utilisé des listes de noms de domaine malveillants fournis par des tiers afin de valider les noms de domaine malveillants détectés. Nous avons vérifié le reste des noms de domaine manuellement en sollicitant les moteurs de recherches.

Nous avons calculé le nombre de noms de domaine malveillants, le nombre de faux positifs détectés, et la vitesse d'analyse du trafic par rapport à la taille minimale pour la création d'une communauté λ et la période de corrélation pour la construction de communautés, voire figures 3.6, 3.7, 3.8 et 3.9.

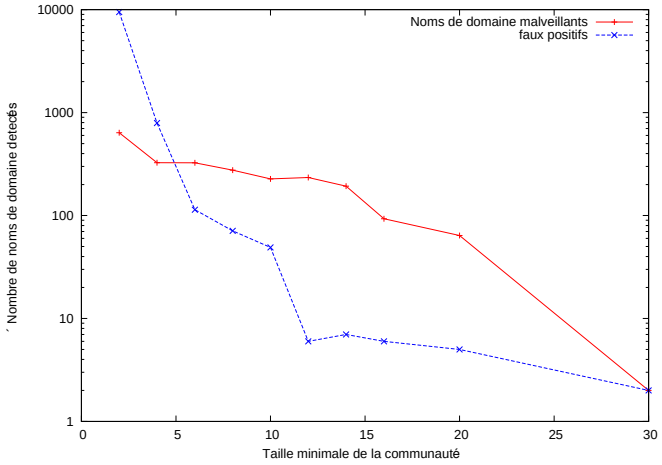

FIGURE 3.6 – Noms de domaine détectés par rapport à la taille minimale d'une communauté

Les valeurs optimales de λ et de la période de corrélation sont le compromis entre un nombre élevé de noms de domaine malveillants détectés, un nombre minimum de faux positifs, et une vitesse d'analyse de trafic optimale.

Le nombre de noms de domaine malveillants est inversement proportionnel par rapport à la taille minimum pour la création d'une communauté. Nous détectons 632 quand la valeur de λ est 2 et 2 noms de domaine malveillants quand la valeur de λ est 30. En revanche, le nombre de noms de domaine détectés décroit légèrement lorsque la valeur de λ est compris entre 10 et 14. En parallèle, le nombre de faux positifs décroit rapidement passant de 9471 à 6 quand λ est inférieur à 12. Au-delà, il se stabilise à moins de 7 faux positifs. En conséquence, la valeur optimale pour la taille minimum pour la création d'une communauté est comprise entre 12 et 14.

Le nombre de noms de domaine malveillants décroit par rapport à la taille minimale de création d'une communauté parce que : plus λ est élevé, moins le système crée de communautés, et moins il détecte de noms de domaine malveillants. De plus, le nombre de faux positifs décroit aussi parce que plus la taille d'une communauté est élevée, plus grande est la probabilité que ses éléments soient membres d'un botnet de type domain-flux, et moins ils génèrent de faux positifs.

La vitesse d'analyse du trafic de notre système de détection décroit très légèrement avec l'augmentation de la taille minimale pour la création d'une communauté λ, 3.8. Puisque moins de communautés sont créées quand la valeur de λ augmente, moins de machines sont rajoutées dans des communautés lors de la construction de communautés. Ainsi, moins de machines seront supprimées de la liste des machines suspectées d'être infectées. En conséquence, plus de machines restent dans cette liste et plus d'opérations de comparaison sont effectuées, qui ralentit l'analyse.

Le nombre de noms de domaine malveillants détectés croît quand la période de corrélation est inférieure à 150 secondes, allant de 31 à 234 noms de domaine. Ce nombre reste stable malgré quelques fluctuations quand la période de corrélation est comprise entre 180 et 600, comme illustré dans la figure 3.8. En parallèle, le nombre de faux positifs détectés est inférieur à 7 quand la période de corrélation est inférieure à 180, ce qui re-

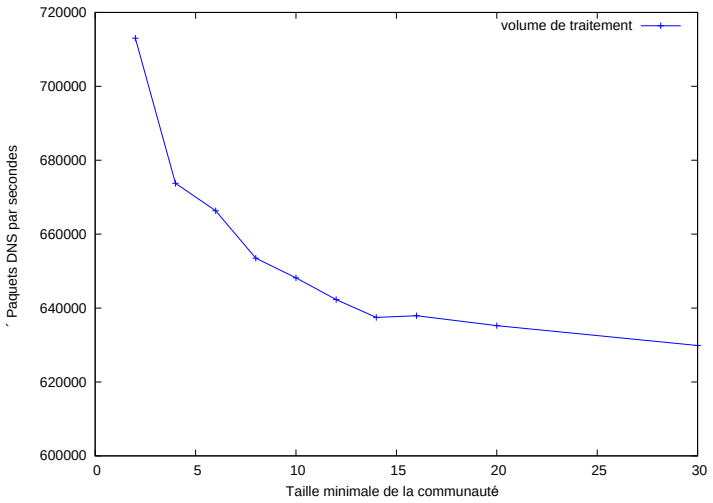

FIGURE 3.7 – Vitesse d'analyse de notre système par rapport à la taille minimale d'une communauté

présente 2% du nombre de noms de domaine détectés. Lorsque la période de corrélation dépasse 180, le nombre de faux positifs se stabilise au-dessus de 50. En conséquence, la valeur optimale de la période de corrélation est de 150 secondes.

La variation de la période de corrélation a un impact direct sur la vitesse d'analyse du trafic dans notre système. Quand la période de corrélation augmente, la vitesse d'analyse du trafic décroit, voir figure 3.9. Ceci parce que quand la période de corrélation augmente la taille de la liste de machines suspectées d'être infectées à analyser augmente, ce qui augmente le nombre de comparaisons de filtres de Bloom.

Enfin, le compromis entre une vitesse d'analyse optimale, un nombre élevé de noms de domaine malveillant détecté, et un taux de faux positifs faible est observé avec une période de corrélation de 150 secondes et une taille minimum de création de communautés λ de 12.

Nous avons détecté lors l'analyse de la capture de 2009, 256 noms de domaine de serveurs de contrôle et commande (C&C) actifs (voir tableau 3.3). Sur ce nombre, 141 noms de domaine ont été générés algorithmiquement par différents botnets. La génération était sur le Second Level Domain (SLD) pour 112 noms de domaine malveillants associés au botnet Conficker, et au Third Level Domain (3LD) pour 29 noms de domaine malveillants associés au botnet Kraken. Un extrait des noms de domaine détectés est représenté dans le tableau 3.4.

TABLE 3.3 – Noms de domaine malveillants

Botnets	Noms de domaine malveillants
Conficker	112
Kraken	29
Bagle	56
Unknown	30

Nous avons aussi détecté 56 noms de domaine reliés au botnet Bagle, qui n'est pas

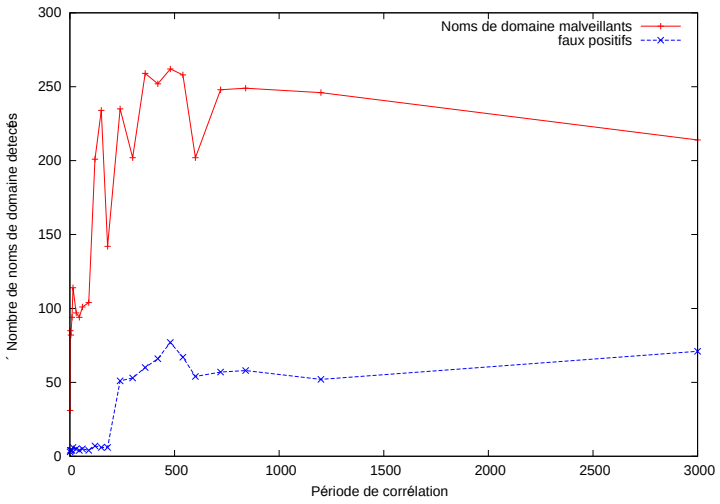

FIGURE 3.8 – Noms de domaine détectés par rapport à la période de corrélation

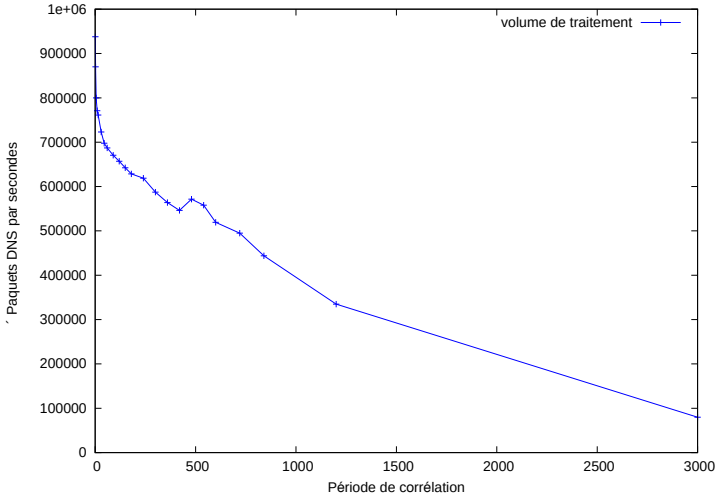

FIGURE 3.9 – Vitesse d'analyse de notre système par rapport à la période de corrélation

un botnet de type domain-flux. Il utilise une liste statique de plusieurs noms de domaine malveillants afin de contacter les serveurs de contrôle et commande. Nous détectons ce botnet parce que plusieurs de ses noms de domaine étaient inaccessibles au moment de la capture, ce qui a entrainé un comportement similaire à celui d'un botnet de type domain-flux.

TABLE 3.4 – extrait de noms de domaine détectés

Conficker	Kraken	Bagle
mvabywjv.org	kuhzahg.dynserv.com	batory.home.pl
prnmxhtas.info	dfyyyopbyzf.dynserv.com	beaupersonalstyling.be
psbgosfbo.ws	mcotackq.mooo.com	billard-88.ch
sjfekashupy.org	icixemu.mooo.com	banyuls.dk
ehazbe.ws	pnpcbmfvhui.mooo.com	bazarmusic.com
ilkhgdb.info	oadcqaqr.dynserv.com	beautywoman.sk
vswfdzdmkfu.info	momktjnclgk.mooo.com	biorem.it

Lors de notre analyse de la trace de 2010, nous avons détecté 49 noms de domaine de serveur de contrôle et commande (C&C) avec un taux de faux positifs de 4%. Sur les 49 noms de domaine détectés, 4 ont été générés algorithmiquement et associés au botnet Conficker, 24 au botnet Bagle, le restant à différents botnets.

Nous avons également analysé une trace réseau d'une heure capturée en 2012 à proximité d'un serveur DNS d'un opérateur réseau à large échelle contenant plus de 180000 utilisateurs. Dans cette capture, le taux de machines infectées par le botnet de type domain-flux Conficker est de 2%. De plus, le taux de machines du réseau qui génère une requête vers un nom de domaine inexistant est de 18%. Par conséquent, la probabilité qu'une machine génère une requête DNS vers un nom de domaine inexistant soit infecté par ce type de botnet est de 11%.

3.5 Conclusion et analyse

Dans ce chapitre, nous avons proposé une nouvelle méthode de détection de botnet conforme aux contraintes d'analyse de trafic dans un réseau à large échelle. Notre méthode de détection identifie les membres d'un botnet de type domain-flux ainsi que leur serveur de contrôle et commande (C&C) tout en préservant la vie privée des utilisateurs du réseau

L'avantage principal de notre méthode de détection est qu'elle analyse le trafic réseau d'un opérateur à large échelle en temps réel. Elle se base sur des structures de données probabilistes qui permettent d'optimiser l'espace mémoire du système de détection. Ces structures de données permettent de garantir la préservation de la vie privée des utilisateurs du réseau.

Notre méthode de détection identifie les noms de domaine malveillants rapidement parce qu'elle ne se base pas sur l'historique des machines ou de celui des noms de domaine demandés. Elle identifie les membres de botnet de type domain-flux ainsi que leurs serveurs associés en se basant seulement sur le comportement des membres du réseau. Afin de s'intégrer à n'importe quel réseau quelque soit leurs spécificités (langues, pays...), notre méthode de détection ne se base pas sur des listes statiques pour filtrer les noms de domaine populaires.

Afin de valider notre système de détection, nous l'avons implémenté afin d'analyser des captures de trafic DNS d'un large opérateur réseau. Nous avons détecté plus de 256 noms de domaine malveillants de serveur de contrôle et commande (C&C), avec une vitesse d'analyse moyenne largement supérieure à la vitesse de capture. Ceci rend notre système adapté à une implémentation en temps réel.

Les membres d'un botnet sont distribués dans plusieurs domaines réseau. Notre méthode requiert un minimum de machines infectées au sein du réseau analysé pour pouvoir construire les communautés de membres du même botnet et leur serveur de C&C associé. En conséquence, l'analyse du trafic réseau généré par plusieurs domaines permet d'atteindre ce seuil de machines infectées. De plus, ceci permet la détection des infections naissantes qui sont indétectables lors de l'analyse du trafic réseau d'un seul domaine réseau.

Les domaines administratifs ne peuvent pas partager le trafic réseau de leurs membres pour être analysé par une entité centralisée. En conséquence, il faut une nouvelle méthode collaborative afin de prendre en compte les contraintes de la collaboration de différents domaines ainsi que les contraintes d'analyse du trafic réseau d'un opérateur à large échelle.

Chapitre 4

Méthodes de détection collaboratives des botnets

4.1 Étude et analyse des méthodes collaboratives

Plusieurs méthodes de détection de botnets ont été proposées. Elles se basent essentiellement sur l'analyse du trafic réseau généré par un groupe de machines, afin d'identifier celles infectées ainsi que leur serveur de contrôle et commande (C&C). La majorité de ces approches se basent sur une architecture centralisée. C'est-à-dire que l'analyse du trafic s'effectue en un point unique du réseau.

Les canaux de contrôle de botnets se basent essentiellement sur les protocoles IRC et HTTP. Ils sont utilisés par les machines infectées pour communiquer avec leur serveur de C&C. Différentes méthodes de détection de botnets se basent sur l'analyse de ces protocoles pour identifier les machines infectées dans un réseau [GPY+07] [GZL08]. En parallèle, d'autres méthodes se basent sur l'analyse du trafic réseau de la couche de transport (TCP et UDP) [ZLP+11] pour identifier les canaux de contrôle de botnets.

La majorité des botnets utilisent le protocole DNS afin d'ajouter plus de résilience à leur architecture. Ils associent un ou plusieurs noms de domaine à leurs serveurs de contrôle et commande (C&C). Les membres du botnet ne connaissent pas l'adresse IP de leur serveur de contrôle au préalable. Ainsi, pour atteindre ce serveur, les machines infectées envoient des requêtes de résolution au serveur DNS. En réponse, ils reçoivent l'adresse IP de ce serveur.

Le protocole DNS est aussi utilisé pour contourner les méthodes de blocages de serveurs de contrôle mis en oeuvre dans les réseaux. La technique fast-flux permet de dissimuler l'adresse réelle du serveur de contrôle aux machines infectées ainsi qu'aux systèmes de blocage. La technique domain-flux permet de contourner le blocage DNS en générant périodiquement une liste de noms de domaine. L'attaquant réserve au préalable l'un de ces noms de domaine afin de l'associer au serveur de C&C.

Plusieurs méthodes comportementales de détections de botnets basées sur l'analyse DNS [CLLK07] [YR12] ont été proposées. Ces méthodes analysent le comportement d'un réseau afin d'identifier les machines infectées et leurs serveurs de contrôle associés. Les auteurs supposent que trafic DNS généré par ces machines est différents de celui d'une utilisation légitime.

Cependant, les membres d'un botnet sont distribués entre plusieurs pays et domaines administratifs différents [Bar07]. Cette distribution rend difficile la détection de botnets en se basant sur un système centralisé puisque chaque domaine analyse seulement le trafic réseau de ses membres. Ceci représente uniquement une partie des communications

malicieuses du botnet.

Les méthodes de détection précédentes corrèlent les événements générés par les membres d'un réseau afin d'identifier un comportement malveillant correspondant à celui d'un botnet. Par la suite, elles détectent les serveurs de contrôle et commande associés à des botnets. Puisque les opérateurs réseau ne peuvent pas partager le trafic réseau de leurs utilisateurs du fait des contraintes légales et business (voir chapitre 3), les solutions de détection proposées par l'état de l'art peuvent analyser seulement le trafic réseau d'un seul opérateur.

Un système de détection collaboratif entre plusieurs opérateurs réseau constitués de plusieurs sondes analyse plus de trafic par rapport à un système centralisé. Ceci doit permettre la détection de botnets fortement distribués entre plusieurs réseaux qui ne le sont pas par les solutions de détection centralisées. De plus, un système collaboratif peut permettre une détection plus précoce des botnets.

Wang H. et al. [WG09] proposent une architecture collaborative hiérarchique de détection de botnet. Dans cette architecture, les domaines administratifs partagent des informations avec des granularités différentes afin de coordonner la détection. Cependant, des informations échangées par les agents d'analyse de trafic tel que le trafic suspect ne respectent pas les contraintes de protection de vie privée des utilisateurs du réseau.

Locaster et al. [LPKS05] proposent Worminator, un système de détection d'intrusion pair-à-pair (P2P) collaboratif. Ce système propose la détection collaborative de machines qui participent à des attaques de type de scan ou balayage du réseau. Chaque acteur du système collaboratif partage périodiquement des filtres de Bloom qui contiennent les adresses IP des machines d'attaque. Ainsi, les autres acteurs peuvent vérifier si leur réseau a aussi été attaqué par les mêmes machines.

Le partage d'information sous forme de filtre de Bloom [Blo70], structure de donnée probabiliste, permet d'optimiser la bande passante réseau. De plus, ces structures de données permettent de dissimuler les adresses IP des machines d'attaques. Seules les victimes des mêmes machines d'attaques peuvent extraire ces adresses IP des filtres de Bloom.

La majorité des systèmes de détection d'intrusion collaboratifs [VK99] [SBD+91] se basent sur un serveur central afin de collecter et corréler les alertes générées par les différents acteurs. Dans ce genre d'architecture, ce serveur central constitue un tiers de confiance pour les différents domaines administratifs et un point de défaillance du système. Si le point central n'est pas accessible, aucune intrusion n'est détectée.

Dans le contexte d'une coopération entre plusieurs opérateurs réseau, il est impossible de centraliser l'analyse ou la corrélation du trafic à cause des contraintes concurrentielles et légales auxquelles ils sont confrontés. Ces contraintes l'empêchent de partager le trafic réseau de ses utilisateurs avec d'autres opérateurs.

Afin de lever ces limitations, il est nécessaire pour un système collaboratif et inter-opérateurs de détection de botnets d'être complètement distribué, autrement dit, un système sans tiers de confiance pour corréler les événements. Ce système doit garantir la confidentialité des données échangées et être performent pour analyse le trafic réseau d'un opérateur.

4.2 Architecture de notre système

Un système de détection de botnets collaboratif et inter-opérateurs doit pouvoir capturer le trafic réseau, en se basant sur des sondes d'analyse, de plusieurs domaines

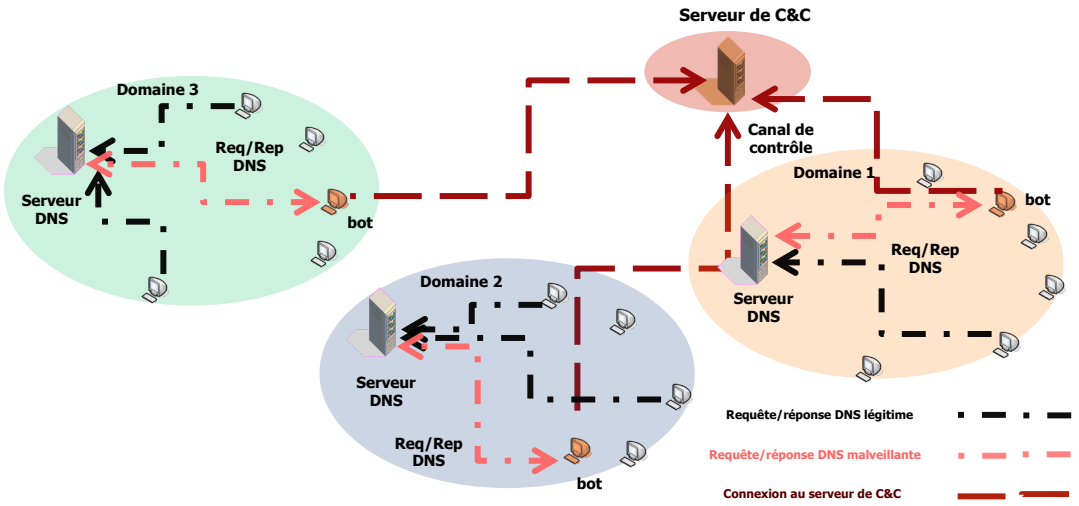

FIGURE 4.1 – Distribution d'un botnet entre plusieurs domaines réseau

réseau. Ces sondes sont interconnectées et distribuées chez les différents domaines. Elles analysent le trafic généré chez chaque domaine tout en échangeant des informations pour coordonner la détection. Ces informations doivent garantir la préservation de la vie privée des utilisateurs et le business de l'opérateur réseau.

La préservation du business de l'opérateur dans notre système signifie qu'une sonde d'analyse de trafic ne doit pas dévoiler le nombre de machines infectées dans le réseau ou la topologie du réseau. Ceci parce que les différents acteurs du système de détection sont des concurrents potentiels.

Puisque les opérateurs réseau du système de détection collaboratif sont des concurrents potentiels, ils ne peuvent pas déléguer la corrélation du trafic chez l'un d'entre eux ou chez un tiers. L'analyse du trafic et la corrélation des événements générés par les membres du réseau doivent être complètement distribuées et réciproques.

Le trafic réseau des membres d'un botnet distribuées dans différents opérateurs converge vers les mêmes serveurs de contrôle et commande (C&C). Les machines infectées, en plus du trafic légitime des victimes ou du trafic d'attaque, génèrent un trafic réseau similaire pour se connecter à leur serveur. Ce trafic est constitué des requêtes et réponses DNS pour atteindre le serveur ainsi que le flux réseau de la connexion vers ce serveur.

Nous avons illustré la distribution d'un botnet entre plusieurs domaines réseau différents dans la figure 4.1. Dans chaque domaine réseau, les membres de ce botnet sollicitent le serveur DNS afin d'obtenir l'adresse IP du serveur de contrôle. Ils établissent ensuite la communication avec le serveur de C&C du botnet.

Un système collaboratif et inter-opérateur de détection de botnets doit pouvoir coordonner l'analyse entre les différentes sondes du réseau afin d'identifier les machines infectées distribuées entre les différents opérateurs et déterminer les points de convergences de leur trafic réseau. Ces points de convergence constituent le trafic généré pour communiquer avec leur serveur de contrôle.

Comme la législation dans la majorité des pays européens restreint le stockage en clair du trafic réseau des utilisateurs d'internet, pour une utilisation commerciale, le processus de détection doit se réaliser en temps réel. Les sondes du réseau doivent pouvoir

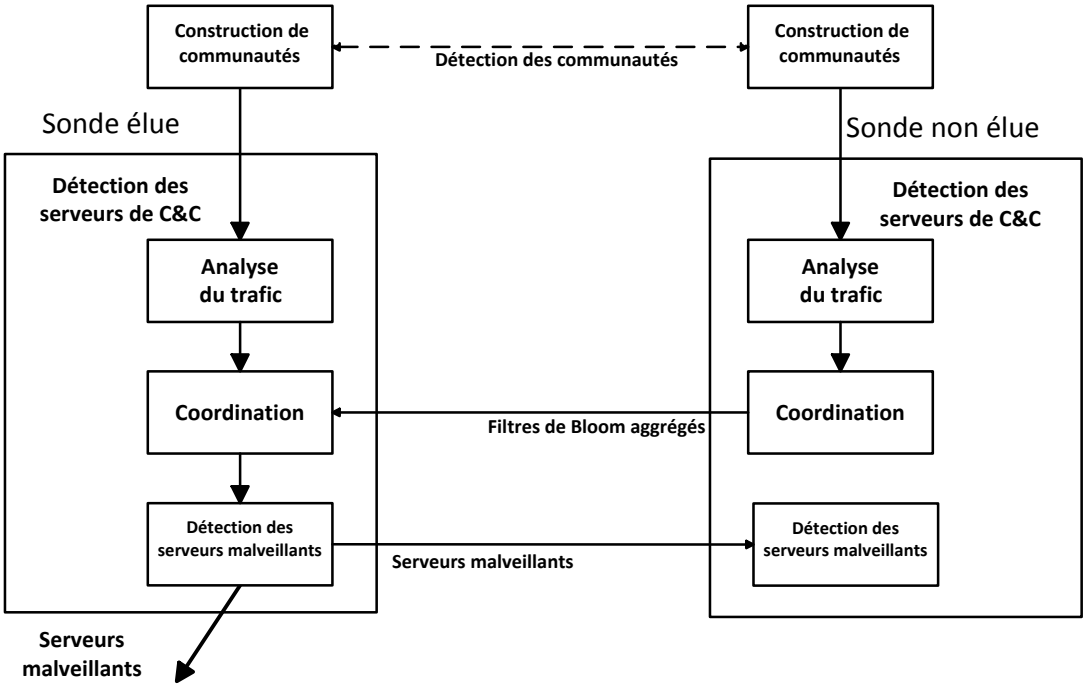

FIGURE 4.2 – Détection collaborative de botnets entre deux acteurs

analyser le trafic et corréler les événements en temps réel d'opérateurs réseau à large échelle.

Nous proposons un système de détection de botnets collaboratif et inter-opérateurs, constitué de plusieurs sondes d'analyse du trafic distribuées dans différents domaines réseau. Les sondes sont interconnectées afin coordonner l'analyse et la corrélation des informations en vue de la détection. Chaque sonde corrèle les informations extraites à partir du trafic capturé avec les informations reçues de la part des autres sondes afin d'identifier les points de convergences correspondants aux activités malveillantes.

Dans notre système de détection collaborative, les sondes d'analyse de trafic sont constituées de deux couches, une couche de construction de communautés et une couche de détection de serveurs de contrôle et commande comme illustré dans la figure 4.2. La couche de détection de communautés identifie les machines infectées et groupe les membres d'un même botnet. La couche de détection de serveurs de contrôle et commande reçoit les informations sur ces communautés et identifie leurs serveurs malveillants associés.

Cependant, la couche de détection de serveurs de contrôle et commande est commune à tous les types de botnets, contrairement à la couche de construction de communauté qui dépend du type de botnet visé par le système de détection. Afin de valider notre méthode collaborative, nous avons implémenté le cas d'usage de botnets de type domain-flux.

4.2.1 Couche de construction de communautés

Notre système de détection collaboratif identifie les membres d'un même botnet et les groupe dans des communautés. Ces communautés peuvent être distribuées entre plusieurs domaines réseau. Cette identification est basée sur la participation dans les

mêmes attaques ou dans la similarité de leur comportement anormal.

Puisque les membres d'une communauté peuvent appartenir à des réseaux différents, différentes sondes peuvent analyser le trafic réseau d'une même communauté. Pour permettre la coordination de la détection et pour réduire le surplus réseau engendré par la collaboration, une sonde est choisie afin de superviser l'analyse du trafic réseau de la communauté.

La sonde élue corrèle les informations reçues de la part des autres sondes ainsi que le trafic des membres de la communauté du réseau supervisé afin d'identifier les points de convergences correspondants aux serveurs de C&C. Par conséquent, le réseau supervisé par cette sonde doit contenir des membres de la communauté afin de pouvoir détecter les serveurs de C&C. Autrement, la sonde élue ne peut pas interpréter le résultat de la corrélation du trafic réseau des membres de la communauté reçus par les autres sondes si elle ne peut pas analyser le trafic d'un membre de la communauté.

Pour garantir l'intégrité de la communauté supervisée, la sonde élue attribue un identifiant à la communauté de machines infectées. Cet identifiant est utilisé par la couche de détection des serveurs de contrôle et commande (C&C) pour permettre la différenciation entre les différentes communautés.

Dans le cas de la détection d'un botnet générant une attaque par déni de service (DDoS), chaque sonde du système peut assigner de manière autonome un identifiant à la communauté de machines infectées construite. Cet identifiant peut être calculé à partir des adresses IP ou des noms de domaine des victimes de l'attaque puisque ces informations sont accessibles aux différentes sondes.

Cette couche de construction de communautés dépend du type de botnet ainsi que du trafic réseau capturé. Elle corrèle le trafic réseau analysé afin d'identifier des similarités comportementales induisant la présence de membres d'un même botnet. Nous allons détailler cette couche pour le cas d'usage de la détection de botnets de type domain-flux.

4.2.2 Couche de détection de serveurs de contrôle et commande

Dans notre système de détection et contrairement à la couche précédente, la couche de détection de serveur de contrôle et commande (C&C) est commune à tous les types de botnets. En effet, quelque soit le type du botnet, le trafic de ses membres convergent vers leur serveur de C&C.

Cette couche reçoit les informations sur les communautés identifiées par la couche de construction de communautés. Ces informations sont reçues uniquement par les sondes qui supervisent des réseaux contenants au moins un membre de ces communautés. Ils contiennent l'identifiant de la communauté, les identifiants de ces membres qui appartiennent au réseau supervisé par la sonde, et l'identifiant de la sonde élue. Cette dernière coordonne l'analyse du trafic réseau des membres de la communauté afin d'identifier les serveurs de commande et contrôle (C&C) du botnet.

Le trafic réseau d'une machine peut seulement être accessible par son opérateur réseau puisqu'il transite en totalité par lui. En revanche, les opérateurs réseau voisins peuvent avoir accès à seulement une partie, correspondant aux paquets transités via ou vers leurs réseaux. En outre, une partie de ce trafic ne quitte pas le réseau de l'opérateur. C'est ainsi le cas d'une machine configurée pour utiliser le serveur DNS récursif de l'opérateur réseau. Son trafic DNS ne quitte pratiquement pas le réseau de l'opérateur puisque c'est le serveur DNS de l'opérateur qui va faire les requêtes récursives.

Les opérateurs ne peuvent et ne veulent pas partager le trafic réseau de leurs clients. Ainsi, afin de permettre une détection collaborative de botnets, chaque sonde doit ana-

lyser le trafic des machines infectées du réseau supervisé afin de détecter leur serveur de contrôle et commande. Afin de coordonner la détection entre les différents réseaux, les sondes échangent des informations anonymisées extraites à partir du trafic de leurs membres.

La couche de détection de serveurs de contrôle et commande est divisée en trois phases : (1) l'analyse du trafic des machines infectées (2) la coordination des sondes d'analyse (3) et la détection des serveurs malveillants. Les différentes phases de détections sont représentées sur la figure 4.2.

Analyse du trafic des machines infectées

Afin de garantir la préservation de la vie privée des utilisateurs du réseau et pour optimiser l'espace mémoire des sondes d'analyse, nous associons un filtre de Bloom avec chaque machine infectée caractérisé par une adresse IP. Le filtre de Bloom est utilisé afin de représenter le trafic réseau des membres d'une communauté identifiée. Afin d'avoir la possibilité de comparer les filtres de Bloom, nous utilisons les mêmes paramètres pour les construire, à savoir la taille du vecteur, le type et le nombre des fonctions de hachage. Par conséquent, si deux machines infectées génèrent un trafic réseau similaire, ils produiront des filtres de Bloom similaires.

Afin que les filtres de Bloom représentent uniquement les dernières activités réseau des machines supervisées et compte tenu de l'impossibilité de la suppression d'une activité, nous remettons à zéro périodiquement leurs vecteurs associés. Cette remise à zéro permet de décroitre le nombre de faux positifs générés à cause du taux de remplissage du filtre de Bloom.

Coordination de l'analyse

Durant cette phase, chaque sonde cordonne l'analyse des membres d'une partie des communautés construites. Pour chacune d'elles, les sondes non élues construisent périodiquement un filtre de Bloom agrégé afin de représenter le point de convergence des activités réseau récentes des membres de ces communautés et qui appartiennent au réseau supervisé par la sonde.

Pour identifier les points de convergence d'une communauté de machines infectées, nous définissons un seuil d'infection. Si le taux de membres de la communauté par rapport à sa taille partagent la même activité réseau, on considère cette activité suspecte.

Le filtre de Bloom agrégé est construit à partir de ceux des membres d'une communauté, voir algorithme 4. Le vecteur associé au filtre de Bloom est constitué d'un ensemble de champ identifiés par un index. Lors de la construction du filtre de Bloom agrégé, nous initialisons un champ du vecteur associé d'index i à 1, si le taux des champs du même index dans les vecteurs des membres de la communauté par rapport à la taille de la communauté dépassent le seuil d'infection. Sinon, nous initialisons le champ d'index i à 0. Par exemple, si la valeur des champs d'index j dans la majorité des vecteurs des filtres de Bloom sont à 1, le champ d'index j dans le vecteur agrégé va être initialisé à 1.

Dans cette phase, le filtre de Bloom agrégé représente les points de convergences de l'activité réseau des membres d'une communauté appartenant au réseau supervisé par la sonde. En effet, seule cette sonde peut analyser le trafic réseau des machines de ce réseau. La construction du filtre de Bloom agrégé permet de garantir la préservation de la vie privée des utilisateurs puisqu'il agrège le trafic réseau de plusieurs. En plus, il

est presque impossible de reconstruire les éléments qui ont permis de remplir le vecteur d'un filtre de Bloom.

Nous définissons également dans notre système une période τ pour que les sondes participant à la détection des serveurs de contrôle d'une communauté de bots, envoient les filtres de Bloom agrégés construits vers la sonde élue. En conséquence, les sondes construisent les filtres de Bloom agrégés des membres de communauté suivant la période τ. Pour réduire la charge réseau nécessaire pour coordonner la détection, les sondes s'échangent les filtres de Bloom agrégés uniquement s'ils ont été modifiés depuis la dernière période. Ainsi, au lieu d'envoyer deux filtres identiques, les sondes envoient qu'un seul.

Algorithm 4 Construction des vecteurs agrégés

```
1: V_A ← 0                                   ▷ Initialiser le vecteur agrégé
2: for i ← 1, N do                           ▷ N : Taille du vecteur
3:     nbr ← 0
4:     for j ← 1, n do                       ▷ n : taille de la communauté
5:         if V_j[i] = 1 then
6:             nbr ← nbr + 1
7:         end if
8:     end for
9:     if nbr ≥ α * n then                   ▷ α : seuil d'infection
10:        V_A[i] ← 1
11:    else
12:        V_A[i] ← 0
13:    end if
14: end for
```

Détection des serveurs malveillants

Cette phase est seulement exécutée par la sonde élue d'une communauté spécifique. La sonde élue analyse les filtres de Bloom des membres de cette communauté qui appartiennent au réseau supervisé par la sonde en plus des filtres de Bloom agrégés reçus. La sonde corrèle ces différentes structures de données et le trafic réseau capturé pour identifier les serveurs de contrôle et commande de botnets.

Quand un membre d'une communauté supervisée par la sonde se connecte à un serveur distant, la sonde vérifie si les autres membres de la communauté se sont également connectés récemment au même serveur. Pour cela, elle teste l'appartenance de la connexion vers ce serveur dans les différents filtres de Bloom des membres de la communauté. Si la majorité des membres de la communauté se sont connectés à ce serveur, on le déclare populaire ou serveur de contrôle et commande du botnet de la communauté.

Nous faisons la différenciation entre les serveurs de C&C et les serveurs populaires de manière dynamique. Nous avons observé lors de notre analyse du trafic que les serveurs populaires légitimes le également sont chez toutes les communautés du système. Par contre, les serveurs de C&C de botnets sont populaires uniquement dans les communautés du botnet considéré.

Chaque sonde est responsable de plusieurs communautés, ainsi elle analyse à chaque moment l'activité réseau de plusieurs communautés. Elle se base sur les filtres de Bloom

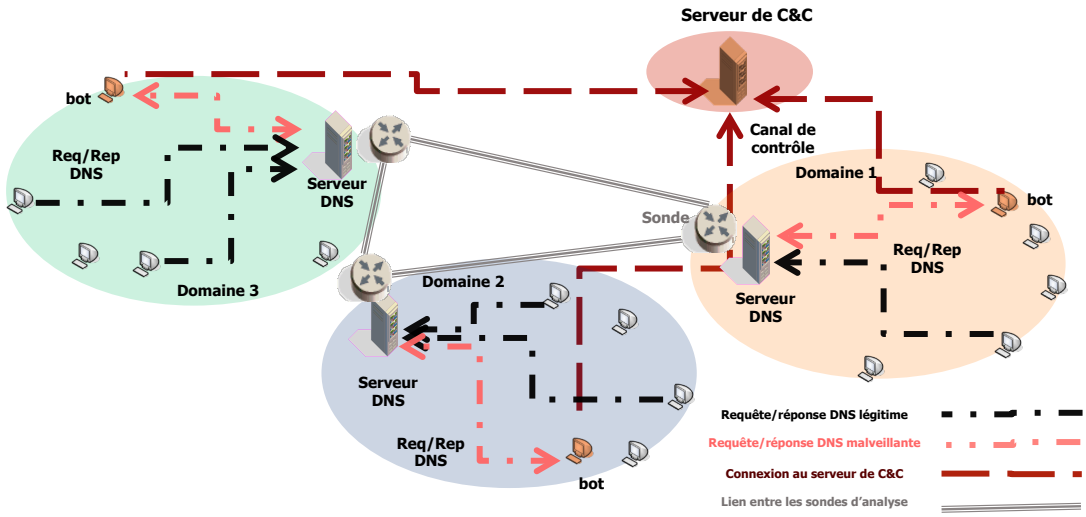

FIGURE 4.3 – Positionnement des sondes d'analyse du trafic DNS pour la détection d'un botnet distribué entre plusieurs domaines réseau

des différentes communautés afin de pouvoir différencier les noms de domaine populaires et les noms de domaine de contrôle et commande (C&C).

Dans notre système, la sonde élue d'une communauté doit contenir au moins un membre de celle-ci. Si ce n'est pas le cas, il lui est impossible d'identifier les serveurs de C&C en se basant uniquement sur les filtres de Bloom agrégés envoyés par les autres sondes.

4.3 Cas d'usage de la détection des botnets de type domain-flux

Afin de valider notre système collaboratif de détection de botnets, nous l'avons mis en oeuvre et évalué pour détecter des botnets de type domain-flux. Le but de notre système de détection est la construction de communautés de machines infectées par le même botnet et l'identification de leur serveur de C&C.

Dans un botnet de ce type, les machines infectées génèrent périodiquement des noms de domaine de manière algorithmique jusqu'à ce qu'ils atteignent leur serveur de C&C qui est réservé au préalable par l'opérateur du botnet. Cette technique permet à ce type de botnets de contourner les solutions de filtrage par listes noires. En effet, ces solutions ignorent quels seront les noms de domaine qui seront utilisés à l'avance par le botnet. Cette technique est utilisée par des botnets connus tels que : Conficker, Zeus, et PushDo.

Afin de pouvoir détecter les botnets de type domain-flux, les sondes de notre système collaboratif sont situées entre les terminaux et les serveurs DNS comme illustrés dans la figure 4.3. Cette position permet de différencier les machines caractérisées par leur adresse IP qui génèrent des requêtes vers les serveurs DNS. Les sondes de notre système capturent et analysent en temps réel le trafic DNS, afin de détecter les serveurs de C&C des botnets.

Chaque sonde de notre système collaboratif accède uniquement au trafic DNS du

réseau qu'elle supervise. Le trafic DNS des machines reste à l'intérieur du réseau du domaine. En conséquence, la collaboration et l'échange d'informations anonymisées entre les différentes sondes du réseau permet aux sondes d'avoir plus d'événements à analyser pour la détection en comparaison à un système se basant sur une seule sonde. La méthodologie suivie pour la détection autonome de botnet de type domain-flux est la même que celle décrite au chapitre 3.

4.3.1 Couche de construction de communautés de machines appartenant à un même botnet

Cette couche détecte et groupe les membres d'un même botnet, distribué entre différents réseaux, en communautés. Les sondes d'analyse positionnées sur ces réseaux échangent anonymement des données afin de pouvoir comparer le trafic réseau analysé.

Dans cette couche, les sondes analysent dans un premier temps les réponses de type NXDomain capturées afin d'identifier les machines infectées et de construire les communautés. Les membres de ces communautés reçoivent des réponses de type NXDomain similaires. En effet, les membres d'un botnet de type domain-flux génèrent des requêtes DNS vers le même ensemble de noms de domaine inexistants.

La construction de communautés est réalisée en deux phases : l'identification des machines suspectées d'être infectées et la construction de communautés.

l'identification des machines suspectes

Cette phase est identique à la méthode décrite dans la détection intra domaine du chapitre 3.3. Le lecteur est invité à se référer à ce chapitre pour plus de détails.

Construction coopérative de communautés

Dans cette phase, nous regroupons les machines suspectées d'être infectées par le même botnet en communautés. Leurs réponses DNS de type NXDomain sont analysées afin d'identifier les membres de la même communauté. Ces membres peuvent être distribuées entre plusieurs domaines réseau et supervisées par des sondes d'analyse différentes.

L'ensemble des noms de domaine inexistants demandés par chaque machine caractérisée par une adresse IP publique est représenté par un filtre de Bloom. Puisque les membres d'un botnet de type domain-flux génèrent des requêtes vers le même ensemble de noms de domaine inexistants, leurs filtres de Bloom associés seront similaires. En conséquence, pour identifier les membres du même botnet, il suffit d'identifier les filtres de Bloom similaires.

Pour permettre cette comparaison, nous associons aux différentes machines du réseau supervisé des filtres de Bloom de même taille. Par ailleurs, cette taille ne doit pas être trop élevée afin d'optimiser l'espace de stockage de ces filtres dans les sondes et de réduire le temps nécessaire à la détection. De plus, cette taille ne doit pas être trop petite pour pouvoir représenter l'ensemble des noms de domaine inexistants demandés par une machine.

Nous avons constaté lors de notre analyse du trafic DNS généré par un opérateur de large échelle qu'une taille de vecteur de filtre de Bloom de 1000 bits est suffisante pour représenter l'ensemble des noms de domaine inexistants généré par la majorité des machines du réseau.

Nous comparons périodiquement les filtres de Bloom des machines suspectes en calculant le taux de similarité, dérivé de la distance de Hamming comme explicité dans le chapitre précédent. L'objectif de cette comparaison est la construction d'un groupe de machines ayant des filtres de Bloom similaires. La construction de ce groupe s'effectue en comparant tous les filtres de Bloom de toutes les machines représentées dans la mémoire de la sonde d'analyse.

Nous définissons la taille minimale d'une communauté par la valeurλ. Si la taille du groupe construit n'atteint pas ce seuil, la sonde ne construit pas de communauté. Cependant, si cette taille dépasse μ, la sonde construit alors une semi-communauté. Elle associe à cette semi-communauté un identifiant afin qu'elle puisse la reconnaitre lors de la corrélation.

Dans cette phase, nous construisons une communauté distribuée entre plusieurs domaines réseau à partir des semi-communautés. Les sondes du réseau s'échangent des informations anonymisées sur leurs semi-communautés construites afin de construire une communauté.

Une semi-communauté est constituée de membres ayant demandé le même ensemble de noms de domaine inexistants. Pour pouvoir comparer entre les semi-communautés en vue de la création d'une communauté, nous associons aux semi-communautés un filtre de Bloom. Ce filtre de Bloom permet de représenter les noms de domaine inexistants demandé par la majorité de ses membres.

La procédure de création de ce filtre de Bloom agrégée est décrite dans l'algorithme 4. Si un champ est à 1 dans la majorité des vecteurs de filtres de Bloom des membres de la semi-communauté, le champ correspondant est initialisé à 1 dans le vecteur du filtre de Bloom agrégé construit. Le filtre de Bloom représente les noms de domaine inexistants demandés par la majorité des membres de cette semi-communauté.

Les sondes de notre système collaboratif comparent entre les filtres de Bloom agrégés afin de pouvoir les regrouper pour créer une communauté de machines appartenant à un même botnet. En effet, si les filtres de Bloom agrégés de deux semi-communautés sont similaires, les ensembles des noms de domaine inexistants demandés par la majorité des membres sont aussi similaires.

Dans notre système collaboratif, les sondes d'analyse échangent des informations sur les semi-communautés crées, afin de pouvoir les regrouper pour construire une communauté. Pour cela, ils échangent l'identifiant de la semi-communauté, l'identifiant de la sonde qui l'a créé, ainsi que le filtre de Bloom agrégé de ses membres.

La construction collaborative d'une communauté se fait en 4 étapes comme illustrées dans la figure 4.4 :

1. Diffusion d'informations : dans cette étape, les sondes diffusent et reçoivent les informations sur les semi-communautés créées.
2. Élection et notification : une sonde identifie les semi-communautés similaires et envoie une notification aux autres sondes qui supervisent ces semi-communautés. Cette sonde devient la sonde élue de l'éventuelle communauté construite.
3. Acquittement : les sondes qui supervisent les semi-communautés répondent à la notification en envoyant un acquittement.
4. Confirmation : la sonde élue envoie la confirmation de création d'une communauté aux autres sondes concernées.

Dans cette phase, lorsqu'une sonde crée une semi-communauté, elle vérifie dans sa mémoire si elle peut la regrouper avec d'autres semi-communautés envoyées par d'autres sondes. Pour cela, elle compare son filtre de Bloom agrégé avec les filtres de

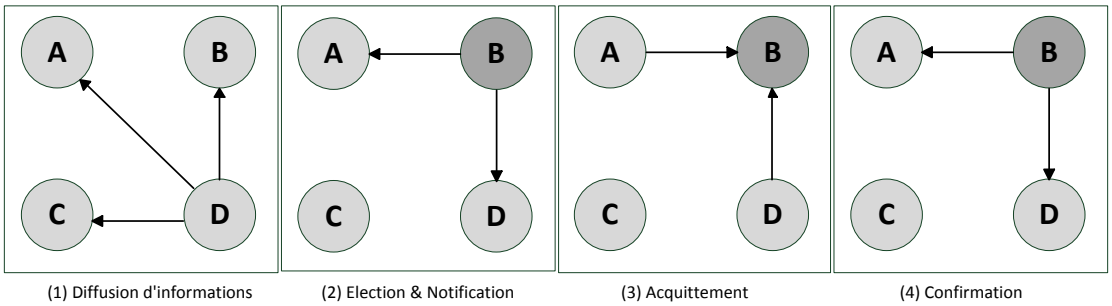

FIGURE 4.4 – Étapes de construction collaborative d'une communauté

Bloom agrégés reçus. Si elle trouve des filtres de Bloom similaires, la sonde envoie des notifications aux autres sondes afin de créer une communauté (étape 2 de la figure 4.4). Cette notification contient les identifiants des semi-communautés attribuées par leur sonde. Par contre, si la sonde ne trouve pas de filtre de Bloom agrégé similaire, elle envoie les informations de la semi-communauté créée vers les autres sondes (étape 1 de la figure 4.4).

Quand une sonde reçoit une notification de création d'une communauté, elle répond avec un acquittement, seulement si elle dispose toujours des informations sur sa semi-communauté (étape 3 de la figure 4.4). En effet, elle vérifie si l'identifiant reçu dans la notification correspond bien à une semi-communauté créée. Ensuite, elle attend la confirmation de création de la communauté.

Si la sonde élue reçoit les acquittements de la part des autres sondes, elle leur envoie une confirmation pour la création de la communauté (étape 4 de la figure 4.4). En parallèle, elle envoie les informations de la communauté et des différentes sondes concernées à la couche de détection de serveurs de contrôle et commande.

Quand une sonde reçoit une confirmation pour la création d'une communauté, elle envoie les informations sur la communauté à la couche de détection de serveurs de contrôle et commande. Pour cela, elle envoie l'identifiant de la communauté, les identifiants des membres supervisés par la sonde et l'identifiant de la sonde élue.

À ce stade de notre travail, nous considérons que tous les acteurs de notre système collaboratif sont fiables. Nous n'avons donc pas abordé le cas où une sonde est compromise. Les conséquences d'une telle hypothèse seront étudiées dans des travaux futurs.

4.3.2 Couche de détection de serveurs de contrôle et commande

Le trafic réseau des membres d'un botnet converge vers leur serveur de contrôle et commande. Afin d'identifier ce point de convergence, on analyse les résolutions DNS réussies de type NOERROR reçues par les membres du botnet. Ce point de convergence correspond au nom de domaine du serveur de contrôle et commande du botnet représenté par la communauté, voir figure 3.5.

Nous associons aux machines infectées et membres de communautés des filtres de Bloom afin de représenter les noms de domaine existants demandés. On extrait le nom de domaine de chaque réponse de type NOERROR, afin de l'ajouter au filtre de Bloom associé à la machine infectée.

Afin d'identifier les points de convergence des noms de domaine existants demandés pas les membres d'une communauté, nous comparons les filtres de Bloom associée aux machines infectées. Par conséquent, nous utilisons les mêmes paramètres pour les filtres

de Bloom (taille du vecteur, nombre et type de fonction de hachage).

L'identification du point de convergence est assurée par la sonde élue de la communauté distribuée entre plusieurs domaines réseau. Pour cela, les autres sondes lui envoient périodiquement les filtres de Bloom agrégés de leurs membres.

Les résolutions DNS réussies par les membres de la communauté supervisés par une sonde non élue sont agrégées afin de mettre en valeur les points de convergence. Un filtre de Bloom agrégé représente les noms de domaine existants demandés par les machines infectées supervisées par cette sonde. Ce filtre agrégé est envoyé périodiquement à la sonde élue de la communauté.

La sonde élue d'une communauté détecte les noms de domaine de serveurs de C&C en se basant sur les filtres de Bloom des machines infectées et membres du réseau supervisé par la sonde. Afin de caractériser les points de convergence du trafic des membres d'une même communauté, nous définissons dans notre système un seuil d'infection. Lorsqu'une machine infectée reçoit une réponse DNS de type NOERROR, la sonde calcule le taux de membres de la communauté qui ont demandé ce nom de domaine. Nous calculons ce taux en vérifiant si les filtres de Bloom associés à ces membres représentent ce nom de domaine. Si ce taux est supérieur au seuil d'infection, nous déclarons que ce nom de domaine correspond à un serveur de C&C, ou à un serveur populaire légitime.

Contrairement aux noms de domaine de serveurs de C&C, les noms de domaine de serveurs populaires légitimes sont sollicités par les membres de toutes les communautés supervisés par la sonde. Par conséquent, nous excluions les noms de domaine sollicités par au moins un membre de la majorité des communautés. Nous déclarons le reste des noms de domaine ceux de serveurs de C&C.

4.4 Évaluations et validations

Afin de valider notre méthode de détection collaborative, nous l'avons implémentée pour le cas d'usage des botnets de type domain-flux entre deux sondes. Les sondes analysent un trafic DNS de 12 heures capturé sur le réseau d'un grand opérateur télécom en 2009 entre les machines et le serveur DNS. Cette capture contient les requêtes et les réponses DNS correspondantes. Cette capture a été anonymisée de telle sorte que les adresses des machines ont été converties en identifiants uniques non réversibles.

Nous avons divisé la capture par rapport aux identifiants des machines du réseau en plusieurs sous-parties pour qu'à tout moment le trafic DNS d'une machine ne soit analysé que par une seule sonde.

Afin de permettre à l'administrateur réseau de contrôler les informations échangées entre les différents domaines administratifs, nous avons utilisé le format IODEF pour encapsuler les messages échangés entre les différentes sondes. Pour représenter un filtre de Bloom dans un message IODEF en utilisant des caractères UTF-8, nous l'encodons en Base64.

L'utilisation du format IODEF lui-même basée sur XML et l'encodage en Base64 des vecteurs binaires des filtres de Bloom ajoutent un surplus dans les informations échangées. En effet, l'encodage Base64 ajoute un surplus de 33% puisque 8 bits nécessaires pour représenter un caractère UTF-8 pour encoder 6 bits. Ainsi, un vecteur de 1200 bits engendre 400 bits de surplus.

Nous avons évalué l'intérêt de la détection collaborative entre deux sondes d'analyse sur la détection autonome. Les performances de la détection de botnets de type domain-

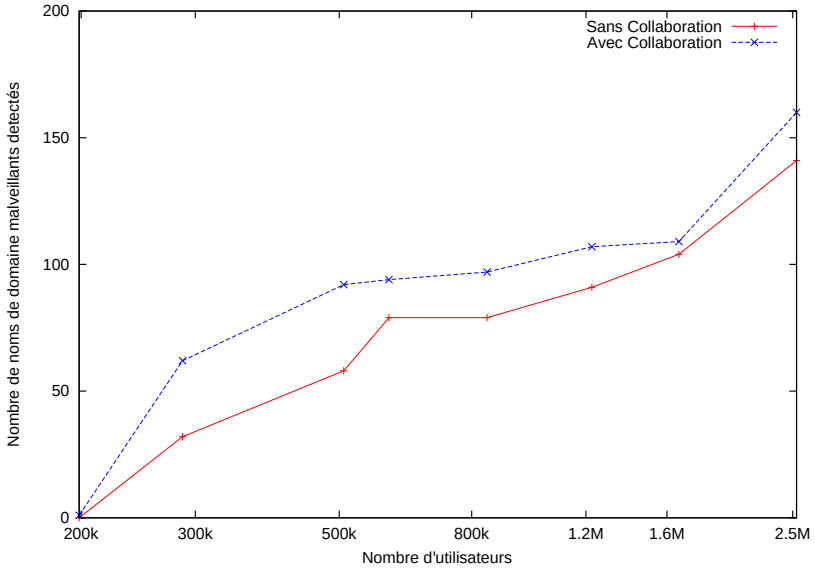

FIGURE 4.5 – Le nombre de noms de domaine malveillants détectés par rapport à la taille du réseau dans la sonde 1

flux autonome ont été décrites dans le chapitre 3.

Afin d'évaluer les bénéfices de l'activation de la détection collaborative dans le cas d'usage d'un botnet de type domain-flux, nous avons mesuré le nombre de noms de domaine malveillants détectés et le nombre de faux positifs dans les deux cas suivants :

– Chaque sonde analyse indépendamment le trafic du sous-ensemble de la capture.
– Les sondes collaborent dans l'analyse du trafic DNS sous-ensemble de la capture.

Nous avons fait varier le nombre de machines du réseau dans les captures analysées par les sondes afin de mesurer les bénéfices de la collaboration, voir les figures 4.5, 4.6, 4.7, 4.8. On voit clairement dans ces figures que plus le nombre de machines du réseau est bas, plus la sonde bénéficie de la collaboration.

Dans la figure 4.5, on remarque que plus le nombre de machines du réseau supervisées par la sonde 1 augmente, moins la collaboration est bénéfique pour cette sonde. Ainsi, quand le nombre de machines est de 300000 le taux de noms de domaine malveillants supplémentaire détecté par la collaboration est de 94%. Ce taux atteint 27% quand le nombre de machines du réseau est de 2.5 millions. On remarque dans la figure 4.6 que le nombre de faux positifs s'accroit avec la collaboration.

Dans la figure 4.7, on remarque que la sonde 2 ne détecte pas de noms de domaine malveillants sans collaboration quand le nombre de machines du réseau analysé est inférieur à 600000. Alors qu'avec le même nombre de machines la collaboration permet à cette sonde d'identifier plus de 80 noms de domaine malveillants. Au-delà de ce nombre de machines, la sonde bénéficie moins de la collaboration. Le taux de noms de domaine supplémentaires détectés par la collaboration est de 20% quand le nombre de machines est de 2.5 millions. Le nombre de faux positifs détectés par cette sonde est supérieur à la sonde 1, voir figure 4.6

On voit clairement que la seconde sonde détecte moins de noms de domaine malveillants que la première. Par conséquent, elle bénéficie plus de la collaboration. Quand

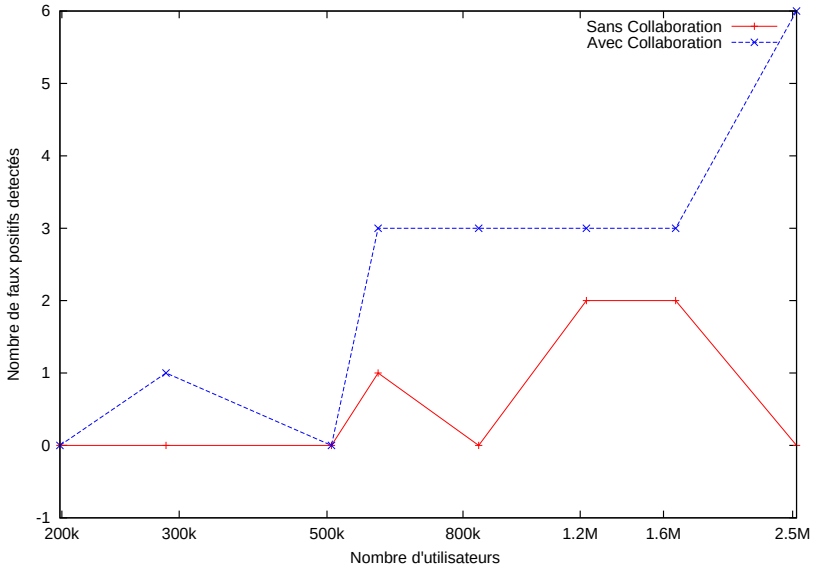

FIGURE 4.6 – Le nombre de faux positifs par rapport à la taille du réseau dans la sonde 1

le nombre de machines des deux sous-ensembles de la capture est de 600000, la sonde détecte 4 noms de domaines malveillants. Alors que la première sonde détecte 80 noms de domaines malveillants. Ceci implique que la deuxième sonde contient moins de machines infectées.

Quand un réseau contient un nombre élevé de machines, il bénéficie moins de la collaboration. Ceci s'explique par le fait que nous utilisons dans cette évaluation, des sous-ensembles originaires d'un seul et même opérateur réseau. Chaque sous-ensemble de la capture contient suffisamment de machines infectées par des botnets de type domain-flux afin de permettre aux sondes de détecter les noms de domaine malveillants sans collaboration.

Les faux positifs identifiés sont des sous-domaines de noms de domaine populaires mentionnés par Alexa. Ils n'ont pas été filtrés de manière dynamique par notre système parce que ces noms de domaine n'ont pas été très populaires parmi toutes les communautés supervisées par notre sonde. Puisque nous nous basons seulement sur le trafic DNS, nous ne pouvons pas affirmer qu'un nom de domaine a été sollicité par le logiciel malveillant ou par la victime de manière légitime. Les logiciels malveillants peuvent ouvrir des connexions vers des sites légitimes pour gêner les systèmes de détection et d'analyse. Par exemple, certaines versions du logiciel malveillant Pushdo [JHKH11] ouvrent plus de 300 connexions vers des sites légitimes.

Nous avons calculé la variation du volume de données réseau générées par les messages échangés entre les sondes dans la couche de construction de communauté et la couche de détection de serveurs de contrôle et commande par rapport à τ, la période de mises à jour de la couche de détection de C&C, voir figure 4.9. Nous avons aussi illustré le volume de données total des messages échangés qui correspond à la somme de ceux générés par les deux couches.

Le volume de données réseau échangées dans les différentes couches décroit par

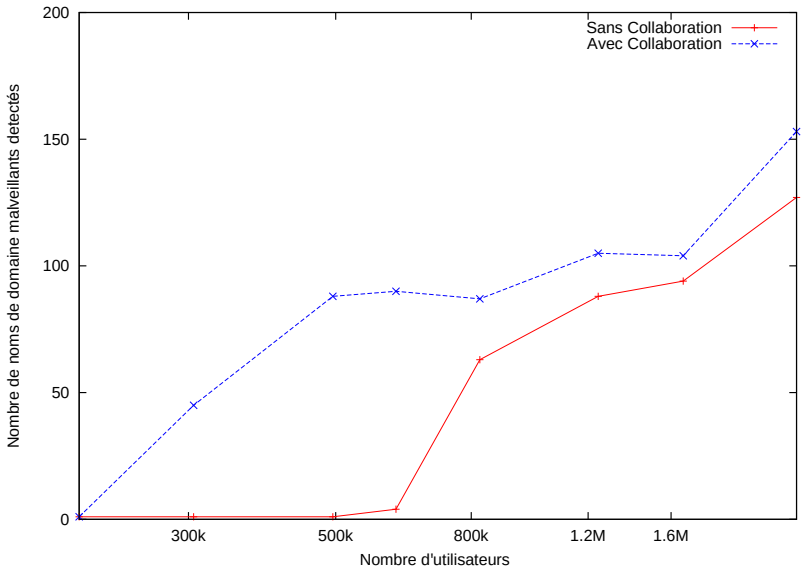

FIGURE 4.7 – Le nombre de noms de domaine malveillants détectés par rapport à la taille du réseau dans la sonde 2

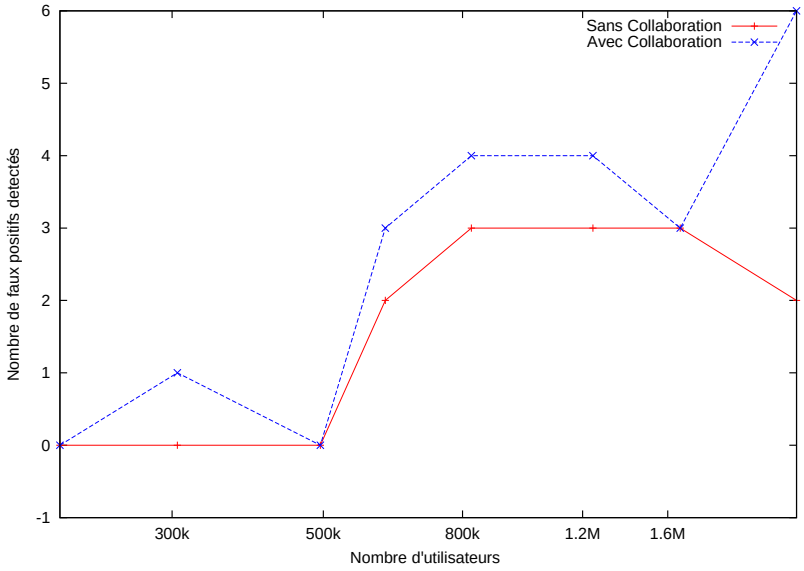

FIGURE 4.8 – Le nombre de faux positifs par rapport à la taille du réseau dans la sonde 2

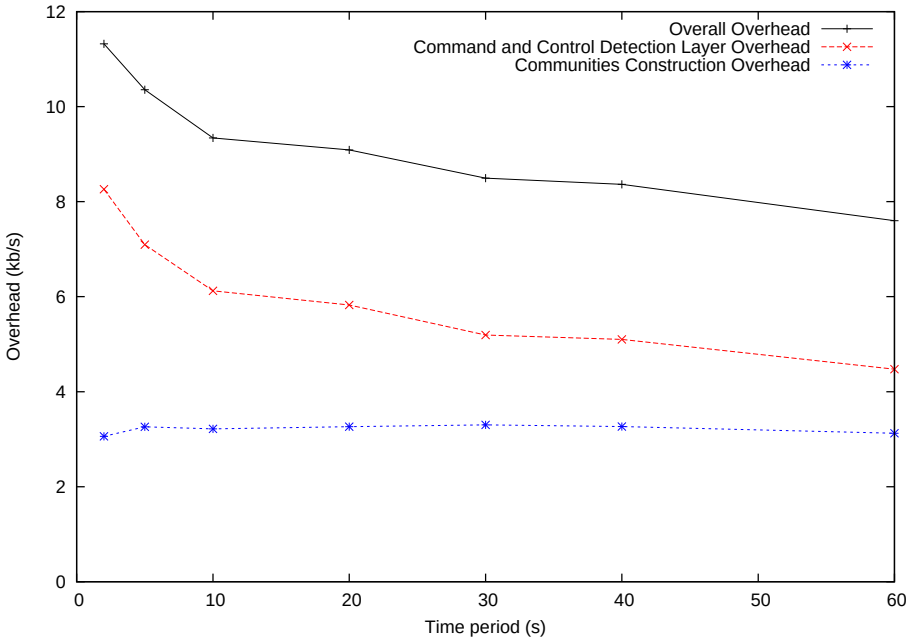

FIGURE 4.9 – Le volume de données généré par notre système

rapport à τ, puisque plus la période est courte, plus les sondes doivent s'échanger des filtres de Bloom agrégés des communautés supervisées. Ces filtres sont encapsulés dans des messages IODEF. Quand la valeur de τ est de 2 secondes, le volume de données échangé est de 11.32 kb/s. Ce volume représente $3.6^{*}10^{-4}$ du trafic réseau global. Ce volume dépend aussi du nombre de communautés supervisées. Ainsi, plus le nombre de communautés est élevé, plus les sondes échangent des messages. En revanche, le volume généré par la couche de construction de communauté est constant parce que la valeur τ n'a pas de répercussion sur cette couche.

Notre implémentation de la détection collaborative de botnets de type domain-flux ne permet pas d'envisager une mise en oeuvre de centaines de sondes du fait du débit nécessaire pour la coordination de l'analyse. Cette méthode est plutôt destinée à une implémentation entre des domaines administratifs puisque chaque sonde peut analyser le trafic d'un réseau contenant plusieurs millions de machines.

Nous pouvons aisément réduire le volume de données généré par les messages de notification diffusés dans la couche de construction de communautés et de la couche d'identification du serveur de C&C par l'utilisation de la compression. Le taux de compression de ces données est très élevé. Il est supérieur à 95% à cause de la répétition des valeurs binaires présentes dans les filtres de Bloom.

Nous avons analysé différents TLD (Top Level Domain ou domaine de premier niveau) des noms de domaine malveillants détectés. Les résultats de l'analyse sont résumés dans le tableau 4.1. Les TLD "ws" et "org" sont les plus utilisées, avec des ratios de respectivement 34% et 33%. Bien que le TLD "com" est le plus utilisé dans la trace de trafic que nous avons analysé, il était présent dans seulement un nom de domaine détecté.

TABLE 4.1 – Distribution des TLD dans les noms de domaine malveillants détectés

TLD	Ratio
WS	0.34
ORG	0.33
INFO	0.20
CN	0.10
COM	0.01
NET	0.01
US	0.01

L'étape de construction des communautés de notre cas d'usage se base sur l'identification de filtres de Bloom similaires représentant le trafic DNS de machines du réseau. Ceci implique une comparaison du contenu éphémère des vecteurs associés aux filtres de Bloom. Cette approche nous empêche d'envisager pour ce cas d'usage une architecture de type DHT (Distributed Hash Table) puisque le contenu des filtres de Bloom varie avec le temps. En revanche, cette approche est parfaitement applicable à d'autres cas d'usage comme celui des botnets de type DDoS puisque l'étape de construction de communauté peut se baser sur des données constantes telles que l'adresse de la cible d'une attaque.

L'activation de la coopération a un impact limité dans la vitesse de traitement de notre système. Nous avons mesuré le nombre moyen de transactions traitées par chaque sonde en activant et en désactivant la collaboration. Elle décroit de seulement 4% quand la collaboration est activée. Ceci est largement compensé par le gain de la collaboration en matière de détection surtout quand le nombre de machines dans le réseau supervisé est inférieur à 1 millions. La vitesse de traitement de chaque sonde est de plus de 500000 transactions par secondes. Ceci dépasse largement la charge du trafic DNS mesurée sur les sous-ensembles des captures Ces résultats permettent d'envisager une implémentation de notre système de détection dans un réseau à large échelle

4.5 Conclusion

Dans ce chapitre, nous avons proposé une méthode de détection collaborative et inter domaine de botnet. Cette méthode est conforme aux exigences de protection de vie privée des utilisateurs, de performance, et de légalité de l'échange d'informations entre des opérateurs réseau. Elle permet d'identifier les communautés de machines infectées par le même botnet et distribuées dans des domaines réseau différents, avant d'identifier leur serveur de commande de contrôle.

Notre système collaboratif de détection est constitué de plusieurs sondes d'analyse dispersées dans des réseaux différents. Ces sondes analysent le trafic réseau et s'échangent des structures de données probabilistes afin de comparer l'activité réseau des machines infectées pour identifier ceux qui font partie du même botnet. Elles s'échangent aussi des structures de données probabilistes pour coordonner la détection des serveurs de C&C des botnets associés aux communautés construites.

Afin de valider notre méthode de détection collaborative, nous avons analysé une capture de trafic anonymisé d'un opérateur réseau. Nous avons démontré que la collaboration de sondes dispersées dans plusieurs réseaux améliore la détection. Ainsi, nous avons 31% de serveurs malveillants supplémentaires lors de l'activation de la collabora-

tion.

Notre méthode de détection collaborative est suffisamment générique pour être appliquée à différents types de botnets. Le processus de construction des communautés peut être basé sur la similitude des attaques. Par exemple pour un botnet de type DDoS, on peut se baser sur la similitude de la cible pour identifier les membres d'un même botnet. De même, dans un botnet de type spam, on peut se baser sur la similitude des messages indésirables pour identifier les membres du même botnet.

L'identification du serveur de contrôle et commande ne suffit pas à stopper la menace liée à un botnet. Ainsi, sans une méthode de réaction appropriée, les machines infectées restent sous le contrôle de l'opérateur du botnet et continuent de générer des attaques.

Chapitre 5

Méthodes réactives aux botnets

5.1 Problématique

La lutte contre les botnets est un enjeu majeur pour les opérateurs réseau du fait de l'impact des attaques générées qui peuvent porter préjudice à leur infrastructure ainsi qu'à leurs clients. En effet, les attaques par déni de service distribuées (DDoS) et les spams génèrent un trafic inutile qui surcharge le réseau de l'opérateur. De plus, la présence de logiciels malveillants dans les machines de leurs clients altère le bon fonctionnement de leur système et ralentit leur vitesse de connexion, dégradant la satisfaction client et entrainant des coûts de support et d'assistance importants.

Les méthodes de réaction contre les botnets consistent essentiellement à bloquer l'accès aux serveurs de contrôle en utilisant des listes noires de noms de domaine ou d'adresses IP de serveurs malveillants préalablement identifiés. Ce blocage permet de rendre le botnet inopérable pour l'attaquant, c'est-à-dire qu'il ne pourra plus communiquer avec les machines infectées.

Pour que le blocage soit efficace, il faut que tous les serveurs de contrôle potentiels du botnet soient inaccessibles. Ainsi, dans le cas d'un botnet de type domain-flux, il ne faut pas qu'un nom de domaine généré puisse être associé au serveur de contrôle. Pour cela, il est ainsi nécessaire de bloquer l'ensemble des noms domaine générés. Ce blocage doit être renouvelé régulièrement selon la période de génération du botnet de type domain-flux. En effet, le blocage de Conficker a impliqué le calcul et blocage de 50000 noms de domaine par jour[LW09].

Dans le cas d'un botnet de type domain-flux, le calcul des noms de domaine qui seront utilisés nécessite de procéder à la rétro-ingénierie du fichier binaire du logiciel malveillant afin de déterminer l'algorithme permettant la génération pseudo-aléatoire de noms de domaine.

L'opération de rétro-ingénierie est très couteuse à mettre en oeuvre pour l'ensemble des botnets actifs. Ainsi, d'après la fondation ShadowServer [sha], il y avait fin mai de l'année 2014 plus de 1700 botnets actifs. De plus, les logiciels malveillants contiennent des mécanismes de chiffrement et d'obfuscation afin de complexifier leur analyse. Par conséquent, la rétro-ingénierie n'est utilisée que pour les botnets contenant un nombre important de machines infectées ou qui ont un impact important.

Les créateurs de botnet ajoutent à leurs logiciels malveillants des mécanismes pour contourner le blocage de leurs serveurs de contrôle et commande (C&C). Ainsi, dans certains botnets lorsque ce dernier n'est pas accessible, les machines infectées essayent de se connecter à des structures décentralisées afin de contourner ce blocage et de rester sous le contrôle de l'attaquant. C'est par exemple le cas d'une variante du botnet

Zeus nommé Gameover Zeus [ARSG+13] qui se base sur une structure décentralisée pour contourner le blocage du C&C.

5.2 Analyse des modes réactifs et contre-mesures

Pour lutter contre les botnets, plusieurs contre-mesures ont été proposées. Nous allons analyser dans cette section : le blocage IP, le blocage DNS, le sink-holing. L'objectif des contre-mesures contre les botnets est de déconnecter les machines infectées de leurs serveurs de contrôle pour qu'elles ne soient plus contrôlées par l'attaquant.

5.2.1 Blocage IP

Le but de cette contre-mesure est d'empêcher les machines infectées d'établir une connexion avec leur serveur de contrôle et commande. Ainsi, si elles ne peuvent pas se connecter à leur serveur de contrôle, elles ne reçoivent pas de commandes d'attaques. De ce fait, elles ne sont plus sous le contrôle de l'opérateur du botnet.

Les routeurs du réseau rejettent les paquets qui sont destinés à l'adresse IP du serveur de contrôle. Les routeurs effectuent ce qui s'appelle le "nullrouting", c'est-à-dire qu'ils envoient les paquets contenant l'adresse IP du serveur de contrôle vers l'interface "null".

Pour appliquer cette contre-mesure, il faut récupérer au préalable l'adresse IP du serveur de contrôle. De plus, il faut s'assurer que cette adresse n'est également pas utilisée par des services légitimes. En effet, il est possible d'héberger plusieurs sites web dans le même serveur. Ainsi, puisqu'ils partagent la même, le blocage IP d'un des sites implique le blocage de tous les sites web hébergés dans le même serveur.

Certains ISP appelés "Rogue ISP" se sont spécialisés exclusivement dans la distribution de contenus malveillants tels que l'hébergement de serveurs de contrôle de botnets [AMJ08], et le phishing [DMPW09]. Ainsi, le blocage des adresses IP de ces ISP permet de perturber les botnets dont les serveurs de contrôle y sont hébergés. Ainsi, le blocage de l'opérateur McColo qui hébergeait les serveurs de contrôle des plus importants botnets spécialisés dans le spam, dont [Shi10] Srizbi, Bobax, Pushdo, et Asprox a entrainé une baisse significative de cette attaque. Dans les premiers jours après son blocage le nombre de spams a baissé de 80% [DMPW09] .

Pour que ce blocage soit effectif, la totalité des adresses IP utilisées par les serveurs de contrôle du botnet doivent être inaccessibles aux machines infectées. Ne pas inclure une adresse IP permet à l'attaquant de l'utiliser pour mettre à jour son botnet et de contourner cette contre-mesure.

Des organismes distribuent des listes noires (blacklists) contenant les adresses IP de serveurs de C&C de botnets. Par exemple, un administrateur réseau peut s'abonner au flux RSS de la fondation Abuse.ch [abu] pour recevoir périodiquement les adresses IP des serveurs de C&C actifs du botnet Zeus.

Puisque cette contre-mesure nécessite la connaissance de l'adresse IP du serveur de contrôle, les attaquants pour la contourner essayent de cacher l'adresse IP de leur serveur. Pour cela, ils utilisent la technique fast-flux [NH08]. Ainsi, en se basant sur le service DNS, les attaquants modifient régulièrement et de manière dynamique les adresses IP reliées aux noms de domaine du serveur de C&C.

Les machines associées à un nom de domaine fast-flux jouent le rôle de proxy inversé entre les bots et leur serveur de contrôle. Le changement dynamique de ces machines

empêche l'application de ce type de blocage. De plus, ces machines correspondent à des utilisateurs qui verraient alors leur connexion internet coupée par une telle contre-mesure. Par exemple, entre le 24 juillet et le 10 septembre 2007 [HGRF08] des chercheurs ont dénombré dans un service de fast-flux plus de 18214 adresses IP uniques partagées entre 814 systèmes autonomes (AS) de différents pays.

5.2.2 Blocage DNS

Afin d'établir une connexion avec leur serveur de contrôle, les machines infectées de la majorité des botnets utilisent le protocole DNS. Possédant le nom de domaine du serveur de contrôle, elles envoient des requêtes vers un serveur DNS pour récupérer son adresse IP. Les botnets utilisent ce protocole pour ajouter plus d'agilité à leur architecture et contourner le blocage IP. Ainsi, si l'adresse IP du serveur de contrôle est bloquée, l'opérateur peut lui en associer une nouvelle.

Dans cette contre-mesure, les résolutions DNS des noms de domaine de serveurs de contrôle et commande (C&C) sont bloquées. Ce blocage DNS peut être mis en oeuvre au niveau du réseau local, au niveau d'un réseau d'entreprise, ou celui d'un opérateur réseau. Comme dans le blocage IP, cette contre-mesure se base sur des listes noires (Blacklists) de noms de domaine de C&C de botnet.

Lors de la réception d'une requête de résolution d'un nom de domaine de C&C, le serveur DNS ne fournit pas l'adresse IP du serveur dans la réponse. À la place, il répond avec une adresse IP invalide comme celle de l'hôte local (localhost) 127.0.0.1, ou il prétend que le nom de domaine n'existe pas. Pour cela, il envoie une réponse spécifique de type NXDomain (nom de domaine inexistant). Par conséquent, les machines infectées ne peuvent pas établir une connexion avec leur serveur de C&C.

Les logiciels malveillants des machines infectées ne sont pas découragés à la réception de ce type de réponses (NXDomain, adresse IP invalide). Ils envoient des requêtes de résolution au serveur DNS jusqu'à ce qu'elles reçoivent une adresse IP valide ou que les machines soient désinfectées. Ces demandes de résolutions des machines infectées surchargent le réseau et le serveur DNS.

Le blocage DNS agit efficacement contre les botnets qui se basent sur la technique fast-flux. En effet, le changement permanent d'adresses IP associées au nom de domaine de contrôle contourne le blocage IP, mais ne permet pas de contourner le blocage DNS puisqu'il n'est pas affecté par ces changements d'adresse.

Pour ajouter plus d'agilité à leur botnet, les attaquants associent plusieurs noms de domaine à leurs serveurs malveillants . Ainsi, si la résolution DNS d'un nom de domaine n'est pas réussie, les machines infectées essayeront de résoudre les autres jusqu'à l'établissement d'une connexion avec un serveur accessible. À titre d'exemple, le botnet Waledac utilisait 277 noms de domaine [wal].

Si un nom de domaine de C&C n'est pas inclus dans la liste de blocage, les machines infectées demeurent sous le contrôle de l'opérateur du botnet. De plus, ce dernier pourra mettre à jour son botnet afin d'utiliser des noms de domaine ou des méthodes de communications alternatifs.

Pour contourner cette contre-mesure, les opérateurs de botnet utilisent la technique domain-flux. Les machines infectées génèrent périodiquement une liste de noms de domaine dont un seul est utilisé par l'attaquant qu'il associe au serveur de C&C . Ainsi, le blocage DNS ne sera efficace que si l'on est en mesure de prévoir l'ensemble des noms de domaine qui seront utilisés par le botnet.

L'automatisation de l'opération de remplissage des listes noires utilisées dans le blocage DNS, implique statistiquement l'insertion de faux positifs. En effet, il est impossible de vérifier manuellement la totalité des noms de domaine inclus dans une liste. De plus, certains algorithmes DGA génèrent des collisions avec des noms de domaine légitimes. Ce risque est d'autant plus élevé que les logiciels malveillants peuvent générer des requêtes vers des noms de domaine légitimes [JHKH11] pour entraver leur analyse, et que le canal de contrôle de certains botnets se base sur des services légitimes. En effet, l'attaquant peut utiliser les réseaux sociaux comme Twitter et Facebook [KMXS10] pour envoyer ses commandes.

5.2.3 Sink-holing

Les méthodes précédentes peuvent être mises en oeuvre uniquement au sein de réseaux supervisés : réseau local, réseau d'entreprise, ou réseau d'opérateur. En revanche, le sink-holing peut avoir un impact sur l'ensemble des machines infectées du botnet considéré.

Dans cette contre-mesure, les machines infectées sont redirigées vers un serveur contrôlé appelé "Sinkhole", lorsqu'elles essayent de se connecter avec leur serveur de C&C. Cette redirection peut être effectuée à deux niveaux : DNS ou IP. Au niveau DNS, les machines infectées reçoivent l'adresse IP du Sinkhole à la place de celle du serveur de contrôle en réponse à leur requête de résolution DNS. Par contre, au niveau IP les paquets envoyés en direction de l'adresse IP du serveur de C&C sont redirigés par les routeurs du réseau vers le Sinkhole.

La redirection DNS vers le Sinkhole est mise en oeuvre selon deux modes : le premier mode est seulement appliqué à un réseau supervisé, et le deuxième mode est appliqué au réseau internet.

- Les requêtes DNS pour les résolutions de noms de domaine des serveurs de C&C sont interceptées au niveau du serveur DNS du réseau supervisé. Ce dernier remplace, dans la réponse DNS l'adresse IP du serveur de C&C du botnet par celle du Sinkhole.
- Le nom de domaine du serveur de C&C est directement associé au Sinkhole par les chercheurs auprès du bureau d'enregistrement DNS (Registrar). Ainsi, toutes les machines infectées par ce botnet essaieront d'établir une connexion avec le Sinkhole en croyant interagir avec leur serveur de C&C. Cette approche nécessite la disponibilité du nom de domaine de C&C ou la collaboration du bureau d'enregistrement DNS.

Certains bureaux d'enregistrement DNS ne sont pas très réactifs aux demandes de blocage ou de redirection de noms de domaine formulées par les chercheurs et les autorités légales des autres pays. Par exemple, le bureau d'enregistrement DNS chinois exigerait plusieurs documents écrits pour procéder au traitement de ces demandes [Bra12]. Ce retard peut permettre aux opérateurs du botnet de changer le nom de domaine ou d'architecture de leur serveur de C&C.

Dans les botnets de type domain-flux, les chercheurs peuvent réserver les noms de domaine qui seront utilisés par le botnet [SGCC+09]. Pour cela, ils font de la rétro-ingénierie sur le logiciel malveillant du botnet afin de déterminer l'algorithme de génération de noms de domaine (DGA). Par exemple, dans le cas du botnet de type domain-flux Conficker, 50000 noms de domaine étaient réservés par jours par le groupe qui coordonnait son démantèlement CWG (Conficker Working Group) [G+11].

Le but de cette contre-mesure est de faire apparaitre aux machines infectées le Sin-

khole comme étant leur serveur de C&C, de collecter des informations sur le botnet, et d'interagir le moins possible avec ces machines afin de ne pas générer d'attaques ou d'être passible de poursuites judiciaires. Pour cela, elles collaborent avec les autorités légales au cours de l'opération.

Dans cette contre-mesure, les chercheurs peuvent recueillir différentes informations sur les machines infectées [LS] : leurs adresses IP, l'horodatage de leur tentative de connexion, leur emplacement géographique. Ces informations peuvent être recueillies seulement en capturant les paquets reçus sans établir de connexion avec les machines infectées. Pour ces dernières, ceci équivaut à un blocage IP puisque leurs paquets ne sont pas acquittés.

Cependant, des chercheurs utilisent des Sinkholes plus actifs qui acceptent l'établissement d'une connexion au niveau de la couche TCP. Ils peuvent ainsi collecter plus d'informations sur le botnet tel que les URL des fichiers malveillants. De plus, ces Sinkholes peuvent recueillir les informations dérobées aux victimes envoyées par les logiciels malveillants [SGCC+09] : les numéros de carte crédits, identifiants de messageries, etc.

En revanche, les chercheurs ne doivent pas envoyer des commandes aux machines infectées. En effet, la police néerlandaise a reçu 55 plaintes à cause de l'envoi d'un programme qui demande le téléchargement d'une solution d'antivirus aux victimes du botnet Bredolab [Dit12].

Pour contourner cette redirection [Bra12], le botnet DNSChanger a configuré ses machines infectées pour qu'elles se connectent à des serveurs DNS sous son contrôle. Ainsi, ces dernières ne recevaient plus l'adresse IP du Sinkhole lors de la résolution DNS. En conséquence, les autorités légales ont détourné le trafic envoyé vers ces serveurs DNS malveillants vers un Sinkhole au niveau IP.

Pour être efficace, les contre-mesures par blocage DNS doivent inclure la totalité des noms de C&C pour les rediriger vers le Sinkhole. En effet, il suffit d'une seule omission pour que l'opérateur du botnet puisse reprendre le contrôle de ses machines infectées. Ainsi, les opérateurs du botnet Mariposa ont pu reprendre le contrôle de leur botnet en corrompant un employé d'un bureau d'enregistrement DNS [NAP+13] pour enregistrer un seul nom de domaine.

En effet, le cout élevé de cette contre-mesure ne permet pas de l'appliquer à tous les botnets à cause du faible retour sur investissement. Ce cout contraste avec la facilité avec laquelle un botnet peut être mis en oeuvre. De fait, ça ne se justifie que pour des botnets contenant des centaines de milliers de machines infectées ou qui génèrent un nombre important d'attaques.

5.3 Proposition de contre-mesures

La remédiation contre les botnets est devenue une préoccupation majeure pour les opérateurs réseau, à cause de l'impact des attaques générées et de contraintes légales éventuelles. L'objectif de notre contre-mesure est de fournir à un opérateur réseau le moyen de remédier à cette menace au sein de son réseau, tout en limitant l'efficacité des techniques de contournement de contre-mesures incluses dans les logiciels malveillants.

Nous proposons une méthode de remédiation qui doit être capable de remédier à un botnet au niveau du réseau sans avoir d'information au préalable sur son fonctionnement. Ainsi, elle doit être suffisamment générique pour s'appliquer à la majorité des botnets, quel que soit le chiffrement utilisé ou la syntaxe des messages échangés. En effet, notre

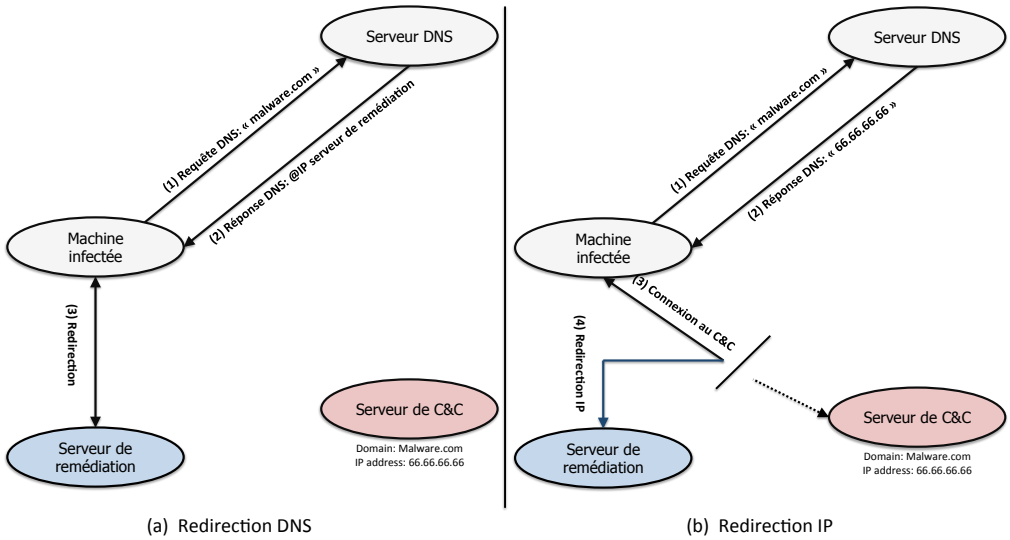

FIGURE 5.1 – Les deux approches de redirection de trafic vers le serveur de remédiation

méthode doit pouvoir remédier à un botnet caractérisé uniquement par l'adresse IP ou le nom de domaine de son serveur de C&C.

Notre serveur de remédiation peut enregistrer les données envoyées par les machines infectées pour des investigations futures. Cependant, puisque ces dernières peuvent contenir les informations personnelles dérobées aux victimes, leur stockage doit être effectué en collaboration avec les autorités légales [Bra12].

Comme le Sinkhole, notre serveur de remédiation interagit directement avec les machines infectées en se faisant passer pour leur serveur de C&C. Cette interaction vise à leurrer les logiciels malveillants et les empêcher de se connecter à d'autres serveurs ou d'utiliser des protocoles alternatifs, et d'éviter que l'opérateur du botnet reprenne le contrôle des machines infectées.

Nous pouvons rediriger les demandes de connexion malveillantes des machines infectées vers notre serveur de remédiation selon deux approches : redirection DNS, et redirection IP. Ces deux redirections sont illustrées dans la figure 5.1.

- Redirection DNS : Dans cette approche, on redirige les requêtes de résolution DNS des noms de domaine de serveurs de C&C vers notre serveur de remédiation.
- Redirection IP : Dans cette approche, les routeurs du réseau redirigent le trafic envoyé en direction de l'adresse IP du serveur de C&C vers notre serveur de remédiation.

Le canal de contrôle de la majorité des botnets est basé sur les protocoles applicatifs IRC et HTTP, qui sont à leur tour basés sur le protocole de la couche transport TCP. Ainsi, pour communiquer avec leur serveur malveillant, les machines infectées établissent une connexion en 3 étapes "Three-Way handshake", avant de pouvoir s'échanger des informations. Dans le cas d'un botnet de type HTTP, les machines infectées envoient des requêtes "HTTP GET" afin de récupérer des informations du serveur ou "HTTP POST" pour envoyer des données dérobées à la victime.

Pour que les machines infectées croient que notre serveur de remédiation est leur serveur de C&C, on doit permettre l'établissement d'une connexion TCP. Par contre puisque les botnets chiffrent leur communication et que nous ne disposons pas de la clé utilisée, la remédiation doit être mise en oeuvre à ce niveau, c'est-à-dire juste après l'établissement de la connexion. En effet, si nous différons la remédiation, les machines infectées

peuvent se rendre compte qu'ils ne sont pas connectés au serveur de C&C soit à cause de l'absence de réponse à leurs requêtes, soit en identifiant un message invalide.

Pour ne pas attirer l'attention des systèmes de détection d'intrusions et des solutions d'antivirus, les logiciels malveillants se basent sur les routines de gestion de connexion du système d'exploitation hôte. Par exemple, en analysant le code source du botnet Zeus [zeud], nous avons constaté qu'il utilise sur les routines du système d'exploitation Windows pour gérer les connexions HTTP. Ainsi, il utilise par exemple la fonction "InternetReadFile" [inta] pour télécharger un fichier du serveur de C&C.

Nous tentons de ralentir la communication entre une machine infectée et notre serveur de remédiation, afin de l'empêcher d'utiliser des serveurs alternatifs ou de passer en mode décentralisé. Pour cela, notre serveur de remédiation essaye de garder la connexion active le plus longtemps possible en exploitant les caractéristiques des routines standards de gestion de connexion du système d'exploitation.

Après l'établissement d'une connexion avec son serveur de C&C, une machine infectée lui envoie une requête pour récupérer les dernières commandes. Dans le cas d'un botnet HTTP, le logiciel malveillant envoie une requête "HTTP GET" et se met en attente de la réponse, comme illustrée dans la figure 5.2. Nous proposons dans notre méthode de remédiation de retarder le plus possible l'envoi de cette réponse, afin de garder le logiciel malveillant en attente.

Pour ne pas attirer l'attention du logiciel malveillant sur notre méthode de remédiation, nous ne devons pas dépasser les différents délais d'expiration ou Timeout mis en oeuvre au sein du protocole TCP et du protocole applicatif utilisé. Ainsi, nous ne retardons pas l'envoi d'acquittement TCP parce que cela ne ferait retarder la connexion que de quelques secondes et le logiciel malveillant renverrait la même requête en croyant que le paquet de la requête a été perdu dans le réseau.

Dans le protocole TCP, une connexion peut être maintenue pendant une longue durée qui dépasse les 2 heures [RB]. En effet, si après son établissement, aucun paquet n'est échangé pendant plus de deux heures la connexion est abandonnée. Par conséquent, dans notre contre-mesure, si le serveur de remédiation n'envoie aucun segment TCP après l'établissement de la connexion, la machine infectée reste en attente.

Cependant, les protocoles applicatifs utilisent d'autres délais d'expiration pour s'assurer du bon fonctionnement de la communication. Ainsi, dans l'implémentation des fonctions HTTP sur le système d'exploitation Windows, un compteur est initialisé après l'envoi d'une requête HTTP avec la valeur de INTERNET_OPTION_RECEIVE_TIMEOUT [intb]. Si aucune réponse n'est reçue avant l'expiration de ce compteur, la requête envoyée est annulée.

La valeur de ce délai d'expiration est fixée par l'application. Ainsi, le logiciel malveillant Zeus fixe la valeur de INTERNET_OPTION_RECEIVE_TIMEOUT à 60 secondes. Par conséquent, notre serveur de remédiation doit répondre aux requêtes de la couche applicative dans un délai qui ne dépasse pas ce Timeout pour garder la connexion active avec le logiciel malveillant.

Cependant, la temporisation de l'envoi de la réponse ne permet de garder la connexion active que pendant la durée de ce Timeout qui est de 1 minute dans le cas du botnet Zeus. Ainsi, après la réception de la réponse, la machine infectée va se rendre compte qu'elle n'est pas en train d'interagir avec son serveur de C&C en examinant la réponse. Cela ne représente pas une différence substantielle par rapport à la contre-mesure de type Sinkhole.

Pour éviter que la machine infectée n'examine la réponse et pour ne pas dépasser le délai d'expiration de connexion, nous proposons à la place de la réponse l'envoi d'un

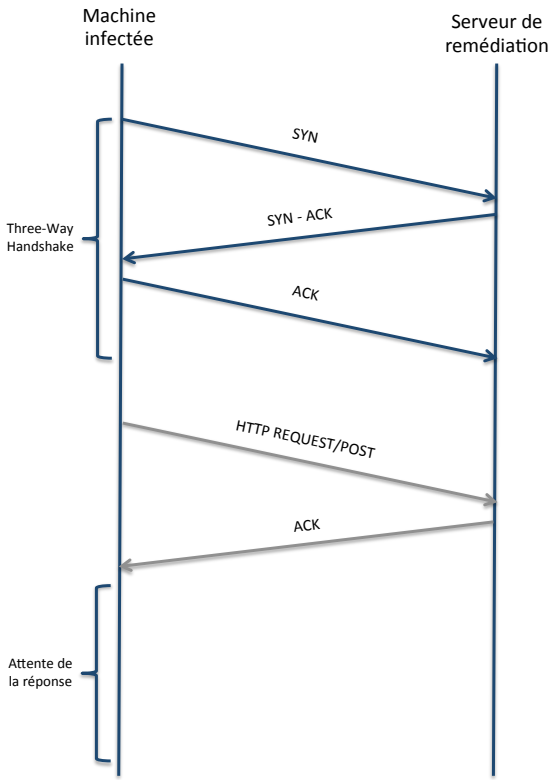

FIGURE 5.2 – Établissement d'une connexion HTTP entre une machine infectée et un serveur de remédiation

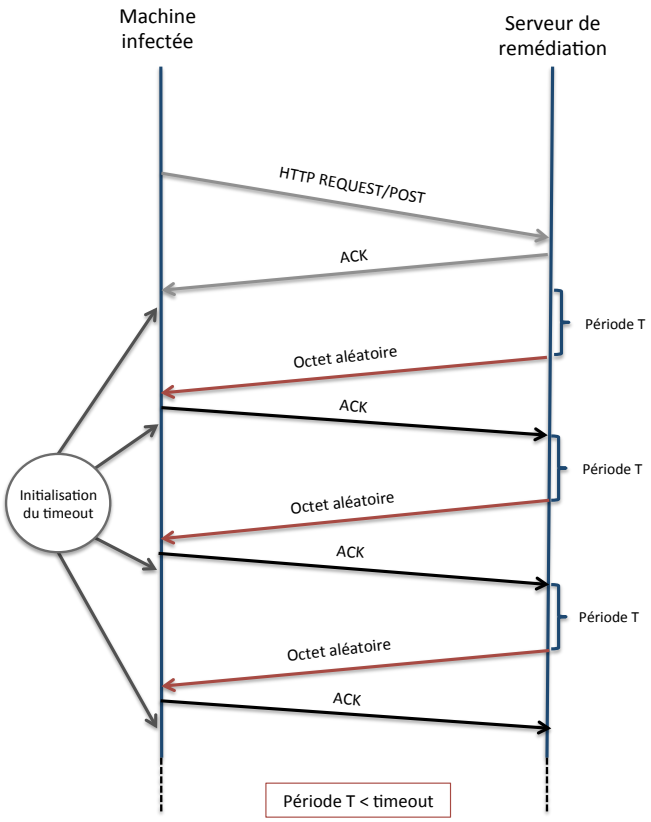

FIGURE 5.3 – Ralentissement d'une connexion HTTP

octet de manière périodique à la machine infectée. Cette période ne doit pas dépasser le délai d'expiration de la connexion pour que le logiciel malveillant garde la connexion active, comme illustrée dans la figure 5.3.

Dans notre contre-mesure, plus la période d'envoi d'octet est longue, moins le serveur de remédiation envoie des données à la machine infectée, et moins la machine infectée détecte rapidement que les données qu'elles reçoivent ne sont pas significatives.

Le logiciel malveillant a besoin de recevoir un volume minimal de donnée pour s'apercevoir que l'hôte avec lequel elle communique n'est pas son serveur de C&C. En effet, il va attendre d'avoir reçu une quantité non négligeable de données avant de les interpréter. Cela est également vrai dans le cas où la communication entre la machine infectée et son serveur de contrôle est chiffrée.

En analysant le code du botnet Zeus, nous avons remarqué que le serveur de ce dernier communique avec les machines infectées par l'envoi de fichiers. Ces dernières interprètent les commandes ou les données reçues seulement après avoir intégralement téléchargé ce fichier. Par conséquent, l'envoi de données invalides par notre serveur de remédiation ne sera détecté par le logiciel malveillant qu'après la fin du téléchargement.

Dans la figure 5.4, nous représentons un extrait du code source de la fonction du téléchargement de données dans le logiciel malveillant Zeus, à partir la ligne 173 du fichier wininet.cpp. On remarque qu'il utilise une boucle infinie pour télécharger les fichiers. Il en sort seulement à la fin de ce téléchargement ou après le dépassement de la taille limite de ce dernier, qui est fixé par le créateur du botnet à 512 kilo-octets.

Lors de la mise en oeuvre de notre méthode de remédiation, le logiciel malveillant Zeus reste bloqué dans cette boucle. En effet, tant que le téléchargement du fichier n'est

pas terminé ou que la taille des données reçues ne dépasse pas 512 kilo octets, le logiciel malveillant reste en attente et n'essaye pas de se connecter à des serveurs alternatifs ou de générer des attaques. Par conséquent, on peut garder la connexion avec le logiciel malveillant Zeus pendant : $2^{19} * T$, où T correspond à la période de génération de l'octet par le serveur de remédiation. Ainsi, pour une valeur de T de 10 secondes, on peut garder la connexion active avec ce logiciel malveillant pendant plus de 60 jours.

Lorsque notre serveur de remédiation envoie un octet à la machine infectée, il active le drapeau PUSH [rfc] pour que la couche TCP du serveur envoie le paquet immédiatement, et pour que la couche TCP de la machine infectée l'envoie directement également à la couche applicative sans passer par le tampon. Cependant, la couche TCP de la machine infectée informe celle du serveur de remédiation que le logiciel malveillant a reçu les données dans un acquittement (ACK) seulement après la réception de l'équivalent d'un MSS (longueur maximum de segment) ou l'équivalent de la moitié du tampon du destinataire. Cela afin d'éviter le phénomène de la fenêtre stupide ou "Silly Window Syndrome" (SWS) [FS11]. Ce mécanisme d'évitement du "Silly Window Syndrome" n'a pas d'impact sur notre méthode de remédiation puisque le serveur de remédiation ne prend pas en compte les valeurs de la taille de la fenêtre de réception de la machine infectée. La longueur maximum de segment (MSS) est négociée lors de l'établissement de la connexion dans les paquets (SYN).

En plaçant le logiciel malveillant dans une position d'attente, notre solution de remédiation permet d'éviter que la machine infectée utilise des serveurs alternatifs ou bascule sur une autre architecture. Ainsi, dans le cas d'un botnet de type domain-flux, le logiciel malveillant ne génèrera pas de nouvelle requête DNS tant que de la remédiation sera effective.

Dans notre méthode de remédiation, nous limitons le blocage aux seules connexions du logiciel malveillant. Ainsi, les logiciels légitimes s'exécutant sur une machine infectée ne seront pas impactés par notre méthode.

5.4 Évaluation et validation

Pour valider notre méthode de remédiation, nous avons déployé un scénario de test comprenant : des routeurs, des machines infectées, un serveur DNS, notre serveur de remédiation, et un Sinkhole. De plus, pour rediriger les machines infectées vers notre serveur de remédiation ou pour bloquer leurs communications, nous avons mis en place un module "DNS Rewriter" permettant de réécrire les réponses DNS des demandes de résolutions de serveurs de C&C, comme illustré dans la figure 5.5.

Pour évaluer l'impact de notre approche, nous nous sommes basés sur le nombre de requêtes DNS générées par le logiciel malveillant. En effet, chaque émission d'une requête de résolution DNS par le logiciel malveillant correspond à une tentative de connexion à un serveur alternatif.

Notre objectif est d'analyser l'efficacité de notre méthode de remédiation sur les logiciels malveillants par rapport aux contre-mesures classiques. Plus précisément, nous souhaitons évaluer son impact sur les communications entre les machines infectées et leur serveur de C&C. Pour cela, nous avons appliqué à chaque machine infectée une contre-mesure différente :

- Blocage NXDomain : Le DNS Rewriter (DNS-RW) répond à toutes les requêtes DNS par une réponse de type NXDomain afin d'empêcher le logiciel malveillant de se connecter au serveur de C&C.

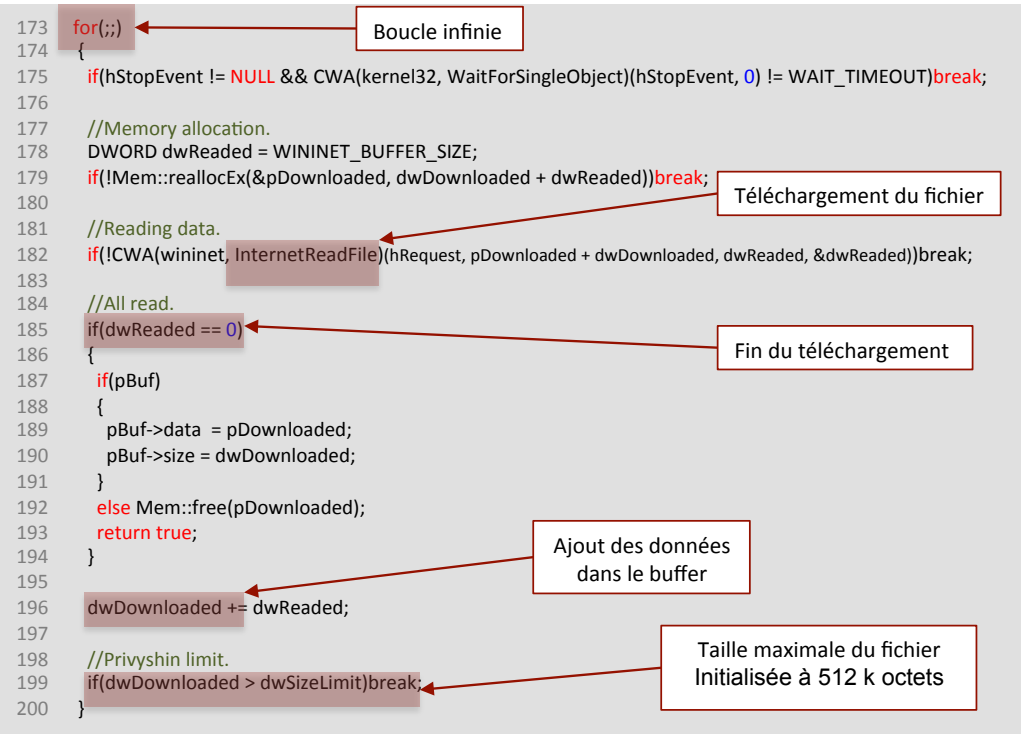

FIGURE 5.4 – Téléchargement d'un fichier dans le logiciel malveillant Zeus

- Blocage DNS : Le DNS-RW renvoie l'adresse IP de l'hôte locale 127.0.0.1.
- Blocage IP : Le DNS-RW envoie dans la réponse, l'adresse IP du Blackhole. Ce dernier supprime les paquets reçus sans générer de réponses.
- Sinkhole passif : Le DNS-RW envoie dans la réponse l'adresse IP du serveur de Sinkhole. Ce dernier n'établit pas la connexion TCP.
- Sinkhole actif : Dans ce cas, le Sinkhole permet l'établissement de la connexion TCP, mais la laisse en attente. Il acquitte les paquets reçus, mais il ne répond pas aux requêtes de la couche applicative.
- Notre contribution : notre serveur de remédiation essaie de ralentir la communication avec le logiciel malveillant en envoyant périodiquement un octet choisi aléatoirement. Dans ces évaluations.

Nous avons analysé le comportement de différents logiciels malveillants pour chacune de ces contre-mesures. Pour chaque test, les six machines étaient infectées simultanément par la même souche de logiciel malveillant. Entre chaque test, les machines étaient réinitialisées afin d'effacer les traces laissées par le logiciel malveillant.

Dans notre évaluation, nous avons installé les logiciels malveillants suivants :

- Ramnit : ce logiciel malveillant est capable d'intercepter les données bancaires des victimes. Il se base sur la technique domain-flux pour localiser son serveur de C&C et utilise un protocole propriétaire au-dessus de TCP pour communiquer avec lui.
- Zeus (Murofet) : spécialisé également dans le vol d'informations bancaires. La version du logiciel malveillant que nous avons évalué utilise un mélange de domain-flux et de P2P pour se connecter à son serveur de C&C qui correspond à la première version du botnet GameOver Zeus [Gam].
- Torpig : aussi appelé Sinowal, est spécialisé dans le vol de données (bancaires, identifiants de messageries..). Il utilise la technique domain-flux pour localiser son serveur de C&C, et le HTTP pour communiquer avec lui.

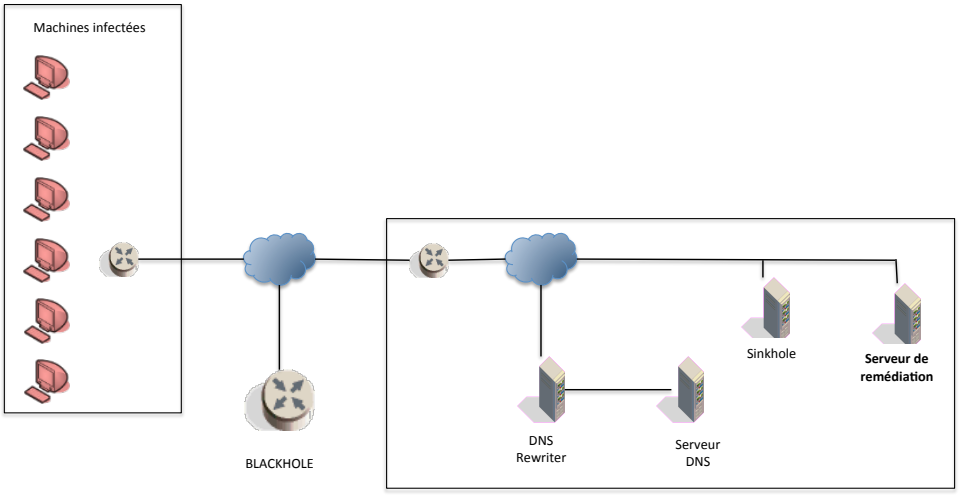

FIGURE 5.5 – Scénario de l'évaluation de notre méthode de remédiation

– Srizbi : est un logiciel malveillant spécialisé dans l'envoi de Spam [Jac]. Il a été responsable pendant une certaine période de la moitié des spams. Il se base sur des adresses IP statiques et le domain-flux pour localiser son serveur de C&C, et HTTP pour communiquer avec lui.
– Kraken : spécialisé également dans le Spam. Il était considéré comme le plus grand botnet en 2008 [Roy08]. Il a été démonté en 2008, mais il est revenu en 2010 en se basant la technique domain-flux pour localiser son serveur de C&C.
– Conficker : a infecté plus de 9 millions de machines. Il se base également sur la technique domain-flux pour atteindre son serveur de C&C. La variante que nous avons analysée génère 50000 noms de domaine par jour.

Le tableau 5.1 résume les résultats de notre évaluation, et permet de comparer l'impact de chaque contre-mesure sur les logiciels malveillants en calculant le nombre de noms de domaine générés.

TABLE 5.1 – Efficacité des méthodes de contre-mesures sur les logiciels malveillants

	Nombre de connexions au C&C tentées par heure					
Type de malware	Blocage			Sinkhole		**Notre contribution**
	NX	DNSBL	IP	Passif	Actif	
Ramnit	**703**	**703**	1964	4107	1302	1303
Zeus (Murofet)	2400	2400	160	2400	113	**< 0.01**
Torpig	14	14	11	14	13	**< 0.01**
Kraken	49450	57600	**183**	194	194	**183**
Srizbi	13	3.2	**0.1**	3.2	3	**0.1**
Conficker.C	62	61	5	21	8	**5**

Les résultats de l'analyse montrent que notre méthode de remédiation permet dans la majorité des cas d'altérer le comportement des machines infectées en réduisant leur capacité de se connecter à leur serveur de C&C.

En acceptant d'établir la connexion avec les bots et de la maintenir active en envoyant périodiquement des données aléatoires, notre méthode de remédiation a donné les meilleurs résultats dans la majorité des logiciels malveillants. Ainsi, dans le botnet

Torpig et Zeus, le logiciel malveillant était complètement paralysé, et ils n'ont pas essayé de se connecter à un autre serveur de C&C durant les 72 heures où ils ont été évalués.

Cependant, les résultats obtenus avec le botnet Ramnit indiquent que notre remédiation ne peut pas être appliquée à tous les logiciels malveillants. En effet, nos tentatives de ralentissement de ses connexions l'ont rendu plus agressif en générant plus de requêtes DNS.

Les résultats des autres méthodes de remédiation sont résumés ci-dessous :

- Blocage DNS : le comportement du logiciel malveillant est globalement similaire lors de la réception d'une réponse de type NXDomain ou de l'adresse IP de l'hôte local (127.0.0.1).
- Blocage IP : puisque le logiciel malveillant se base sur des délais d'expiration de la couche applicative, cette contre-mesure permet de réduire les tentatives de connexions, même si son impact diffère selon le botnet.
- Sinkhole passif : son impact est globalement similaire à celui du blocage IP.
- Sinkhole actif : En acceptant d'établir la connexion et de ne plus interagir ensuite, cette contre-mesure permet de réduire le nombre de connexions puisque le logiciel malveillant reste connecté plus de temps.

5.5 Conclusion

Nous avons proposé dans ce chapitre une nouvelle méthode de remédiation contre les botnets qui permet de réduire drastiquement les tentatives de connexions des machines infectées à leurs serveurs de C&C. Pour cela, elles sont redirigées vers un serveur de remédiation qui essaie ensuite de garder la connexion active le plus longtemps possible en envoyant périodiquement des données aléatoires.

Nous avons démontré que notre méthode est suffisamment générique pour remédier à différents logiciels malveillants. En effet, elle ne nécessite pas de connaitre le type du botnet, la syntaxe des messages échangés, ou la clé de chiffrement utilisée. Elle exploite l'utilisation par les logiciels malveillants des routines de gestions de connexions du système d'exploitation hôte.

Afin de valider notre méthode, nous avons comparé son impact sur les connexions réseau par rapport aux contre-mesures de l'état de l'art. Elle a obtenu des résultats largement supérieurs dans la majorité des cas examinés. En effet, elle a bloqué la communication de deux botnets largement répandus (GameOver Zeus et Torpig) pendant toute la durée de l'analyse, soit 72 heures.

Nous avons évalué essentiellement des botnets basés sur le protocole HTTP. Cependant, notre approche peut être mise en oeuvre pour ralentir les connexions d'autres protocoles, tels qu' IRC.

Chapitre 6

Conclusion et perspectives

6.1 Importance du problème

Les travaux présentés dans ce mémoire de thèse portent sur la problématique des botnets. Cette problématique représente des menaces importantes associés à un impact financier significatif. Il est bien établi que ce problème sera parmi les plus «dangereux» pour la sécurité d'Internet dans le proche avenir, et plusieurs pays se mobilisent déjà conjointement pour tenter de l'enrayer.

6.2 Résultats

Dans ces travaux, de nouvelles approches sont proposées pour lutter de manière plus efficace contre ce fléau :

1. Une nouvelle approche [GSAM11] permettant d'identifier les machines infectées qui participent à des attaques par déni de service avec un nombre de paquets nécessaires largement inférieur aux méthodes proposées dans l'état de l'art, et cela sans générer de trafic réseau supplémentaire.

2. Une approche [GMS13b, GSAM12] de détection des serveurs de contrôle de botnets présentant un faible taux de faux positifs et répondant à des problématiques de coeur de réseau. La vitesse d'analyse mesurée lors de l'implémentation permet d'envisager une implémentation temps réel de notre méthode dans un réseau de type opérateur Telecom.

3. Une approche de détection collaborative [GMS13a] des serveurs de contrôle de botnets dans un contexte inter-opérateur. Nous avons démontré lors de notre implémentation que la collaboration augmente significativement le nombre de serveurs de contrôle détecté par rapport à un système autonome.

4. Une solution de remédiation des botnets [MDG13] permettant d'empêcher les machines infectées de se connecter à leur serveur de contrôle. En effet, l'implémentation de notre méthode démontre qu'elle permet de limiter fortement les tentatives de connexion des logiciels malveillants par rapport aux contre-mesures de l'état de l'art et apporte ainsi une solution concrète dans la lutte contre les botnets.

6.3 Moyens

Pour identifier les machines infectées, nous avons proposé une nouvelle méthode de traçabilité qui permet de remonter à la source d'attaques par déni de service directes et indirectes de type ICMP tel que SMURF. Pour cela, nous nous sommes basées sur le fonctionnement du protocole ICMP pour faire parvenir à la cible les informations des routeurs du chemin de l'attaque.

Pour détecter les serveurs de contrôle et commande (C&C) des botnets dans un réseau à large échelle, nous avons proposé une méthode qui se base sur l'analyse du trafic DNS généré par les membres d'un botnet de type domain-flux. Notre méthode ne se base pas sur les signatures du trafic de logiciels malveillants connus, et repose uniquement sur le comportement anormal des machines infectées par ce type de botnets. Durant l'analyse, notre méthode s'appuie sur des structures de données probabilistes permettant atteindre les objectifs de volumétrie et de confidentialité requis pour envisager une implémentation dans un opérateur réseau.

Puisque les machines infectées d'un botnet sont distribuées entre plusieurs domaines réseaux différents, nous avons proposé une méthode collaborative inter-opérateur de détection de botnet. Notre méthode se base sur la coordination des communications et des attaques des membres d'un même botnet pour les détecter. Pour pouvoir respecter les contraintes d'une collaboration entre des opérateurs différents, nous échangeons des structures de données probabilistes permettant de supprimer toute information sensible des données transmises.

Notre dernière contribution est une méthode de remédiation de botnets qui empêche les machines infectées de se connecter à leur serveur de contrôle et commande (C&C). Pour cela, notre méthode essaie d'établir et de garder une connexion active avec les logiciels malveillants pour les empêcher de se connecter à des serveurs alternatifs. Nous avons démontré qu'elle est suffisamment générique pour remédier à différents botnets.

6.4 Perspectives

Les travaux de cette thèse permettent d'ouvrir des perspectives dans les directions suivantes :

- Les méthodes de traçabilité permettent à la cible de déterminer la localisation des machines d'attaques. Cependant, si aucune remédiation n'est mise en oeuvre au niveau de l'opérateur, le lien réseau de la cible demeure saturé. Ainsi, une réflexion doit être menée pour un mécanisme de communication entre les systèmes de détection et les routeurs de l'opérateur pour appliquer rapidement une remédiation d'une attaque en cours. Ainsi, si un système de détection communique avec les routeurs de l'opérateur réseau de la localisation des attaquants, ce dernier peut mettre en place un filtrage périmétrique efficace des paquets pour que la cible soit accessible aux membres de l'opérateur. De plus, cette réflexion doit impliquer une collaboration entre différents opérateurs pour que la cible reste accessible aux maximums d'utilisateurs légitimes.
- Nous avons défini de manière empirique des seuils pour nos méthodes de détection. Ainsi, nous avons démontré qu'avec les seuils définis, il faut que le réseau analysé contienne au minimum 200000 utilisateurs actifs pour détecter des botnets de type domain-flux. Il serait intéressant de considérer l'utilisation d'une machine d'apprentissage pour configurer dynamiquement ces seuils suivant les spécificités

du réseau. De plus, la modélisation mathématique d'un tel système permettrait de considérer des réseaux de plus petite taille, tels que les réseaux d'entreprises.

– Nous avons démontré que l'activation de la collaboration améliorait le taux de détection des serveurs de contrôle de plus de 31% en moyenne. Dans des travaux futurs, il serait intéressant de comparer les résultats de l'utilisation de traces capturées sur des réseaux différents par rapport à l'utilisation de traces capturées sur le même réseau avec un nombre d'utilisateurs égaux.
– Notre approche, basée sur la comparaison des filtres de Bloom, que nous avons adoptée pour corréler le trafic généré par les machines du réseau peut être utilisée dans d'autres contextes d'analyse de données. Ces dernières peuvent être distribuées dans des emplacements différents. En effet, cela correspond dans nos travaux de thèse à la détermination des membres d'un même botnet distribués dans des domaines réseau différents.
– Pour valider notre méthode collaborative de détection, nous avons implémenté et évalué le cas d'usage des botnets de type domain-flux. Notre système peut théoriquement être utilisé pour détecter les botnets de type DDoS distribués entre plusieurs réseaux. De la même façon, notre méthode de traçabilité du trafic IP peut être utilisée pour identifier les membres d'un même botnet lors d'une attaque DDoS de type ICMP directe ou réflective.
– L'approche de remédiation proposée nous a permis d'obtenir des résultats extrêmement prometteurs lors de nos expérimentations sur différents botnets. Ces résultats restent cependant préliminaires, car évalués sur un échantillon réduit de logiciels malveillants et principalement focalisés sur les flux de contrôle de type TCP. Il serait ainsi nécessaire de poursuivre ces travaux sur d'autres types de logiciels malveillants, et de voir notamment dans quelle mesure ce type de remédiation pourrait être adaptée à des botnets de type P2P par exemple.

Annexe A

Publications de l'auteur

A.1 Conférences internationales

1. Hachem Guerid, Ahmed Serhrouchni, Mohammed Achemlal, Karel Mittig, "A Novel Traceback Approach for Direct and Reflected ICMP Attacks," in IEEE Network and Information Systems Security (SAR-SSI), 2011, pp.1- 5, 18-21, May 2011.
2. Hachem Guerid, Karel Mittig, Ahmed Serhrouchni, "Privacy-Preserving Domain-Flux Botnet Detection in a Large Scale Network", in IEEE 5th International Conference on Communication Systems and Networks (COMSNETS 2013), Bangalore, India, 7-10, January 2013.
3. Hachem Guerid, Karel Mittig, Ahmed Serhrouchni, "Collaborative Approach for Inter-domain Botnet Detection in Large-scale Networks" In Collaborative Computing : Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on (pp. 279-288). IEEE.

A.2 Conférence nationale

1. Hachem Guerid, Karel Mittig, Ahmed Serhrouchni, "Détection de botnets domain-flux dans un réseau à large échelle", Proceedings of the 7th Conference on Network and Information Systems Security (SAR-SSI), ISBN 978-2-9542630-0-7, Cabourg - France, 2012

A.3 Rapports techniques

1. DEMONS D1.2, Architecture Specification - First Draft, DEMONS FP7-257315, 2011.
2. DEMONS D5.2, Design and Specifications of DEMONS Application Adaptation Layer, DEMONS FP7-257315, 2011.
3. DEMONS D5.3, Preliminary Implementation of DEMONS Applications, DEMONS FP7-257315, 2012.
4. DEMONS D1.5, DEMONS architecture specification, DEMONS FP7-257315, 2012
5. DEMONS D5.4, Final Specification and Implementation of DEMONS Applications, DEMONS FP7-257315, 2012

6. DEMONS D7.4, Preliminary Trials results, DEMONS FP7-257315, 2012.
7. DEMONS D7.5, Assessments and Trials results, DEMONS FP7-257315, 2013.

A.4 Brevet

1. Karel Mittig, Nicolas Deschamps et Hachem Guerid : Procédé de ralentissement d'une communication dans un réseau, 12 2013. Institut national de la propriété industrielle, numéro 7062320.

Bibliographie

[ABB13] D ALYIAS, D BATCHELDER et J BLACKBIRD : Microsoft security intelligence report july-december 2012. Rapport technique, Microsoft, 2013.

[abu] The swiss security blog. `http://www.abuse.ch/`.

[ald11] Bargain-basement botnet kit. `http://www.theregister.co.uk/2011/09/22/aldi_bot/`, 2011.

[ale] Alexa top sites. `http://www.alexa.com/topsites`.

[Alj03] H. ALJIFRI : Ip traceback : a new denial-of-service deterrent ? *Security Privacy, IEEE*, 1(3):24–31, 2003.

[AMJ08] Jart ARMIN, J MCQUAID et M JONKMAN : Atrivo–cyber crime usa, 2008.

[APD+10] Manos ANTONAKAKIS, Roberto PERDISCI, David DAGON, Wenke LEE et Nick FEAMSTER : Building a dynamic reputation system for dns. *In USENIX Security Symposium*, pages 273–290, 2010.

[APN+12] Manos ANTONAKAKIS, Roberto PERDISCI, Yacin NADJI, Nikolaos VASILOGLOU, Saeed ABU-NIMEH, Wenke LEE et David DAGON : From throw-away traffic to bots : detecting the rise of dga-based malware. *In Proceedings of the 21st USENIX conference on Security symposium*, Security'12, pages 24–24, 2012.

[ARSG+13] Dennis ANDRIESSE, Christian ROSSOW, Brett STONE-GROSS, Daniel PLOHMANN et Herbert BOS : Highly resilient peer-to-peer botnets are here : An analysis of gameover zeus. *In Malicious and Unwanted Software :" The Americas"(MALWARE), 2013 8th International Conference on*, pages 116–123. IEEE, 2013.

[BA03a] A. BELENKY et N. ANSARI : On ip traceback. *Communications Magazine, IEEE*, 41(7):142–153, 2003.

[BA03b] Andrey BELENKY et Nirwan ANSARI : Ip traceback with deterministic packet marking. *Communications Letters, IEEE*, 7(4):162–164, 2003.

[Bak] F BAKER : Rfc 1812.

[Bar00] C BARROS : A proposal for icmp traceback messages. *ICMP Traceback Working Group*, 2000.

[Bar07] David BARROSO : Botnets-the silent threat. *European Network and Information Security Agency (ENISA)*, 15:171, 2007.

[bet13] "beta bot" marks the latest banking malware to hit the online underground. `http://www.scmagazine.com/beta-bot-marks-the-latest-banking-malware-to-hit-the-online-underground/article/295408/`, 2013.

[BKKB11] Leyla BILGE, Engin KIRDA, Christopher KRUEGEL et Marco BALDUZZI : Exposure : Finding malicious domains using passive dns analysis. *In NDSS*, 2011.

[Blo70] B.H. BLOOM : Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[BLT03] Steven Michael BELLOVIN, Marcus LEECH et Tom TAYLOR : Icmp traceback messages, 2003.

[BM04] A. BRODER et M. MITZENMACHER : Network applications of bloom filters : A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[BOB+10] Hamad BINSALLEEH, Thomas ORMEROD, Amine BOUKHTOUTA, Prosenjit SINHA, Amr YOUSSEF, Mourad DEBBABI et Lingyu WANG : On the analysis of the zeus botnet crimeware toolkit. *In Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, pages 31–38. IEEE, 2010.

[Bra12] Danny BRADBURY : Fighting botnets with sinkholes. *Network Security*, 2012(8):12–15, 2012.

[CA98] CERT Advisory CA : 01 smurf ip denial-of-service attacks. *URL http ://www. cert. org/advisories/CA-1998-01. html*, 1998.

[Car] Carberp source leak. `https://blog.damballa.com/archives/2042`.

[CdQC12] Rodrigo Alves COSTA, Ruy JGB de QUEIROZ et Elmano Ramalho CAVALCANTI : A proposal to prevent click-fraud using clickable captchas. *In Proceedings of the 2012 IEEE Sixth International Conference on Software Security and Reliability Companion*, pages 62–67. IEEE Computer Society, 2012.

[CGKP11] Juan CABALLERO, Chris GRIER, Christian KREIBICH et Vern PAXSON : Measuring pay-per-install : The commoditization of malware distribution. *In USENIX Security Symposium*, 2011.

[CJM05] Evan COOKE, Farnam JAHANIAN et Danny MCPHERSON : The zombie roundup : Understanding, detecting, and disrupting botnets. *In Proceedings of the USENIX SRUTI Workshop*, volume 39, page 44, 2005.

[CLLK07] Hyunsang CHOI, Hanwoo LEE, Heejo LEE et Hyogon KIM : Botnet detection by monitoring group activities in dns traffic. *In Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on*, pages 715–720. IEEE, 2007.

[Coo10] Peter COOGAN : Spyeye bot versus zeus bot, 2010.

[cry] Cryptolocker creeps lure victims with fake adobe, microsoft activation codes. `http://www.theregister.co.uk/2014/01/02/cryptolocker_worm/`.

[CTD+10] Alper CAGLAYAN, Mike TOOTHAKER, Dan DRAPAEAU, Dustin BURKE et Gerry EATON : Behavioral patterns of fast flux service networks. *In System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1–9. IEEE, 2010.

[Dam13] DAMBALLA : Pushdo evolves again : Enhances evasion with domain generation algorithm, 2013.

[dar] Evil new ddos botnet lurking in the darkness. `http://blogs.computerworld.com/17489/evil_new_ddos_botnet_lurking_in_the_darkness`.

[DFS02] Drew DEAN, Matt FRANKLIN et Adam STUBBLEFIELD : An algebraic approach to ip traceback. *ACM Transactions on Information and System Security (TISSEC)*, 5(2):119–137, 2002.

[Dit12] David DITTRICH : So you want to take over a botnet. *In Proceedings of the 5th USENIX conference on Large-Scale Exploits and Emergent Threats*, pages 6–6. USENIX Association, 2012.

[DKP+12] Alberto DAINOTTI, Alistair KING, Ferdinando PAPALE, Antonio PESCAPE *et al.* : Analysis of a/0 stealth scan from a botnet. *In Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 1–14. ACM, 2012.

[DMPW09] Steve DIBENEDETTO, Daniel MASSEY, Christos PAPADOPOULOS et Patrick J WALSH : Analyzing the aftermath of the mccolo shutdown. *In Applications and the Internet, 2009. SAINT'09. Ninth Annual International Symposium on*, pages 157–160. IEEE, 2009.

[dns] Dns-based blachole list. `http://www.dnsbl.info/`.

[egg] Eggdrop development. `http://eggheads.org/`.

[fak] Fake av uses stolen digital certificate to evade detection. `http://www.hotforsecurity.com/blog/fake-av-uses-stolen-digital-certificate-to-evade-detection-7351.html`.

[FCT11] Gregory FEDYNYSHYN, Mooi Choo CHUAH et Gang TAN : Detection and classification of different botnet c&c channels. *In Autonomic and Trusted Computing*, pages 228–242. Springer, 2011.

[FJT+08] Marc FOSSI, Eric JOHNSON, Dean TURNER, Trevor MACK, Joseph BLACKBIRD, David MCKINNEY, Mo King LOW, Téo ADAMS, Marika Pauls LAUCHT et Jesse GOUGH : Symantec report on the underground economy. *Symantec Corporation*, 2008.

[FS] P FERGUSON et D SENIE : Rfc2827 (bcp38) : Network ingress filtering : Defeating denial of service attacks which employ ip source address spoofing. ietf, may 2000.

[FS05] Ahmad FADLALLAH et Ahmed SERHROUCHNI : Denial of service attack and defense schemes analysis and taxonomy. *In 3rd International Conference : Sciences of Electronics Technologies of Information and Telecomm.*, 2005.

[FS11] Kevin R FALL et W Richard STEVENS : *Tcp/ip illustrated*, volume 1. Addison-Wesley Professional, 2011.

[G+11] Conficker Working GROUP *et al.* : Conficker working group : Lessons learned, 2011.

[Gam] Fbi disrupts gameover zeus and cryptolocker botnet. `https://www.abuse.ch/?p=7822`.

[GBC+12] Chris GRIER, Lucas BALLARD, Juan CABALLERO, Neha CHACHRA, Christian J DIETRICH, Kirill LEVCHENKO, Panayiotis MAVROMMATIS, Damon MCCOY, Antonio NAPPA, Andreas PITSILLIDIS *et al.* : Manufacturing compromise : the emergence of exploit-as-a-service. *In Proceedings of the 2012 ACM conference on Computer and communications security*, pages 821–832. ACM, 2012.

[GH07] Jan GOEBEL et Thorsten HOLZ : Rishi : Identify bot contaminated hosts by irc nickname evaluation. *In Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 8–8. Cambridge, MA, 2007.

[GMS13a] Hachem GUERID, Karel MITTIG et Ahmed SERHROUCHNI : Collaborative approach for inter-domain botnet detection in large-scale networks. *In Collaborative Computing : Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*, pages 279–288. IEEE, 2013.

[GMS13b] Hachem GUERID, Karel MITTIG et Ahmed SERHROUCHNI : Privacy-preserving domain-flux botnet detection in a large scale network. *In Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pages 1–9. IEEE, 2013.

[GN12] Maria GARNAEVA et Yury NAMESTNIKOV : Ddos attacks in h2 2011. *Kaspersky Securelist*, 2012.

[Gol11] Steve GOLD : Taking down botnets. *Network Security*, 2011(5):13–15, 2011.

[GPY+07] Guofei GU, Phillip PORRAS, Vinod YEGNESWARAN, Martin FONG et Wenke LEE : Bothunter : Detecting malware infection through ids-driven dialog correlation. *In Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, page 12. USENIX Association, 2007.

[GPZ+08] Guofei GU, Roberto PERDISCI, Junjie ZHANG, Wenke LEE *et al.* : Botminer : Clustering analysis of network traffic for protocol-and structure-independent botnet detection. *In USENIX Security Symposium*, pages 139–154, 2008.

[GSAM11] Hachem GUERID, Ahmed SERHROUCHNI, Mohammed ACHEMLAL et Karel MITTIG : A novel traceback approach for direct and reflected icmp attacks. *In Network and Information Systems Security (SAR-SSI), 2011 Conference on*, pages 1–5. IEEE, 2011.

[GSAM12] Hachem GUERID, Ahmed SERHROUCHNI, Mohammed ACHEMLAL et Karel MITTIG : Détection de botnets domain-flux dans un réseau à large échelle. *In Network and Information Systems Security (SAR-SSI), 2012 Conference on*, 2012.

[GSN+07] Julian B GRIZZARD, Vikram SHARMA, Chris NUNNERY, Brent ByungHoon KANG et David DAGON : Peer-to-peer botnets : Overview and case study. *In Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 1–1, 2007.

[GZL08] G GU, J ZHANG et W LEE : Botsniffer : Detecting botnet command and control channels in network traffic. *In Proceedings of the 15th Annual Network and Distributed System Security Symposium, NDSS*, 2008.

[Hac10] Guerid HACHEM : Analyse et traçabilité des attaques smurf. Mémoire de D.E.A., UPMC, 2010.

[HGRF08] Thorsten HOLZ, Christian GORECKI, Konrad RIECK et Felix C FREILING : Measuring and detecting fast-flux service networks. *In NDSS*, 2008.

[HKZHH13] Fariba HADDADI, H Gunes KAYACIK, A Nur ZINCIR-HEYWOOD et Malcolm I HEYWOOD : Malicious automatically generated domain name detection using stateful-sbb. *In Applications of Evolutionary Computation*, pages 529–539. Springer, 2013.

[Hol05] Thorsten HOLZ : A short visit to the bot zoo [malicious bots software]. *Security & Privacy, IEEE*, 3(3):76–79, 2005.

[HSD+08] Thorsten HOLZ, Moritz STEINER, Frederic DAHL, Ernst BIERSACK et Felix C FREILING : Measurements and mitigation of peer-to-peer-based botnets : A case study on storm worm. *LEET*, 8:1–9, 2008.

[IAB14] Price Water House Coopers IAB : 2003 internet advertising revenue full-year report, 2014.

[ICA] List of top-level domains. `http://data.iana.org/TLD/tlds-alpha-by-domain.txt`.

[IH05] Nicholas IANELLI et Aaron HACKWORTH : Botnets as a vehicle for online crime. *CERT Coordination Center*, 1:28, 2005.

[inta] Internetreadfile function. `http://msdn.microsoft.com/en-us/library/windows/desktop/aa385103%28v=vs.85%29.aspx`.

[intb] Option flags. `http://msdn.microsoft.com/en-us/library/windows/desktop/aa385328.aspx`.

[IRC] Irc is dead, long live irc. `http://royal.pingdom.com/2012/04/24/irc-is-dead-long-live-irc/`.

[Jac] Kelly JACKSON : Srizbi botnet sending over 60 billion spams a day.

[JCJ+10] Nan JIANG, Jin CAO, Yu JIN, Li Erran LI et Zhi-Li ZHANG : Identifying suspicious activities through dns failure graph analysis. *In Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 144–153. IEEE, 2010.

[JHKH11] Gregoire JACOB, Ralf HUND, Christopher KRUEGEL et Thorsten HOLZ : Jackstraws : Picking command and control connections from bot traffic. *In USENIX Security Symposium*, 2011.

[KBMA+09] Ethan KATZ-BASSETT, Harsha V MADHYASTHA, Vijay K ADHIKARI, Arvind KRISHNAMURTHY et Thomas ANDERSON : Practical reverse traceroute. NANOG, 2009.

[KDG+12a] E. KOSTA, J. DUMORTIER, H. GRAUX, R. TIRTEA et D. IKONOMOU : Sandvine. Rapport technique, Sandvine, 2012.

[KDG+12b] E. KOSTA, J. DUMORTIER, H. GRAUX, R. TIRTEA et D. IKONOMOU : Study on data collection and storage in the eu. Rapport technique, ENISA, European Network and Information Securiy, 2012.

[kel] Microsoft halts another botnet kelihos. `http://www.cnet.com/news/microsoft-halts-another-botnet-kelihos/`.

[Ken96] Malachi KENNEY : Ping of death. *Insecure. org*, 1996.

[KGSD07] Turgay KORKMAZ, Chao GONG, Kamil SARAC et Sandra G DYKES : Single packet ip traceback in as-level partial deployment scenario. *International Journal of Security and Networks*, 2(1):95–108, 2007.

[KMXS10] Erhan J KARTALTEPE, Jose Andre MORALES, Shouhuai XU et Ravi SANDHU : Social network-based botnet command-and-control : emerging threats and countermeasures. *In Applied Cryptography and Network Security*, pages 511–528. Springer, 2010.

[KRH07] Anestis KARASARIDIS, Brian REXROAD et David HOEFLIN : Wide-scale botnet detection and characterization. *In Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, volume 7. Cambridge, MA, 2007.

[Kum07] S. KUMAR : Smurf-based distributed denial of service (ddos) attack amplification in internet. *In Internet Monitoring and Protection, 2007. ICIMP 2007. Second International Conference on*, pages 25–25, 2007.

[KWU12] Seung Hyun KIM, Qiu-Hong WANG et Johannes B ULLRICH : A comparative study of cyberattacks. *Communications of the ACM*, 55(3):66–73, 2012.

[Ler13] Michèle LERUTH : Cryptolocker, une e-menace bien inquiétante !, 2013.

[Les07] Michael LESK : The new front line : Estonia under cyberassault. *Security & Privacy, IEEE*, 5(4):76–79, 2007.

[Lin09] P. LIN : Anatomy of the mega-d takedown. *Network Security*, 2009(12):4–7, 2009.

[LLC13] Hui-Tang LIN, Ying-You LIN et Jui-Wei CHIANG : Genetic-based real-time fast-flux service networks detection. *Computer Networks*, 57(2):501–513, 2013.

[Loe12] Victoria LOEWENGART : *An introduction to Hacking and Crimeware : A Pocket Guide*. 2012.

[LPKS05] Michael E LOCASTO, Janak J PAREKH, Angelos D KEROMYTIS et Salvatore J STOLFO : Towards collaborative security and p2p intrusion detection. *In Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 333–339. IEEE, 2005.

[LRG11] Wei LU, Goaletsa RAMMIDI et Ali A GHORBANI : Clustering botnet communication traffic based on n-gram feature selection. *Computer Communications*, 34(3):502–514, 2011.

[LRST00] Felix LAU, Stuart H RUBIN, Michael H SMITH et Ljiljana TRAJKOVIC : Distributed denial of service attacks. *In Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 3, pages 2275–2280. IEEE, 2000.

[LS] Rainer LINK et David SANCHO : Lessons learned while sinkholing botnets–not as easy as it looks ! *In Proceedings of the 21st Virus Bulletin International Conference,(Barcelona)*, pages 106–110.

[LSXL04] Jun LI, Minho SUNG, Jun XU et Li LI : Large-scale ip traceback in high-speed internet : Practical techniques and theoretical foundation. *In Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 115–129. IEEE, 2004.

[LW09] F. LEDER et T. WERNER : Know your enemy : Containing conficker. *The Honeynet Project, University of Bonn, Germany, Tech. Rep*, 2009.

[LWLS06] Carl LIVADAS, Robert WALSH, David LAPSLEY et W Timothy STRAYER : Using machine learning technliques to identify botnet traffic. *In Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 967–974. IEEE, 2006.

[LWM09] Felix LEDER, Tillmann WERNER et Peter MARTINI : Proactive botnet countermeasures–an offensive approach. *The Virtual Battlefield : Perspectives on Cyber Warfare*, 3:211–225, 2009.

[LXG+09] Jing LIU, Yang XIAO, Kaveh GHABOOSI, Hongmei DENG et Jingyuan ZHANG : Botnet : Classification, attacks, detection, tracing, and preventive measures, 2009.

[MDG13] Karel MITTIG, Nicolas DESCHAMPS et Hachem GUERID : Procédé de ralentissement d'une communication dans un réseau, 12 2013. Institut national de la propriété industrielle numéro 7062320, Patent Pending.

[MMW+01] Allison MANKIN, Dan MASSEY, Chien-Lung WU, S Felix WU et Lixia ZHANG : On design and evaluation of. *In Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on*, pages 159–165. IEEE, 2001.

[MR04] Jelena MIRKOVIC et Peter REIHER : A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[NAP+13] Yacin NADJI, Manos ANTONAKAKIS, Roberto PERDISCI, David DAGON et Wenke LEE : Beheading hydras : performing effective botnet takedowns. *In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 121–132. ACM, 2013.

[Naz07] Jose NAZARIO : Blackenergy ddos bot analysis. *Arbor*, 2007.

[Naz08] Jose NAZARIO : Georgia ddos attacks-a quick summary of observations. *arbor Sert (Security engineering and response team)*, 12, 2008.

[NH08] Jose NAZARIO et Thorsten HOLZ : As the net churns : Fast-flux botnet observations. *In Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 24–31. IEEE, 2008.

[OR] J OIKARINEN et D REED : Rfc 1459 : Internet relay chat protocol, may 1993. *Status : EXPERIMENTAL*.

[OR93] Jarkko OIKARINEN et Darren REED : Internet relay chat protocol. 1993.

[PAG12] Roberto PERDISCI, Davide ARIU et Giorgio GIACINTO : Scalable fine-grained behavioral clustering of http-based malware. *Computer Networks*, 2012.

[Pax01] Vern PAXSON : An analysis of using reflectors for distributed denial-of-service attacks. *ACM SIGCOMM Computer Communication Review*, 31(3): 38–47, 2001.

[PGP12] D. PLOHMANN et E. GERHARDS-PADILLA : Case study of the miner botnet. *In Cyber Conflict (CYCON), 2012 4th International Conference on*, pages 1–16, June 2012.

[PLF10] Roberto PERDISCI, Wenke LEE et Nick FEAMSTER : Behavioral clustering of http-based malware and signature generation using malicious network traces. *In NSDI*, pages 391–404, 2010.

[Pos81] Jon POSTEL : Internet control message protocol. 1981.

[Pow13] POWERTECH : Smurf amplifier registry, oct 2013.

[Pri13] Matthew PRINCE : The ddos that knocked spamhaus offline (and how we mitigated it). 2013.

[PSM+10] Aiko PRAS, Anna SPEROTTO, Giovane MOURA, Idilio DRAGO, Rafael BARBOSA, Ramin SADRE, Ricardo SCHMIDT et Rick HOFSTEDE : Attacks by anonymous wikileaks proponents not anonymous. 2010.

[RB] Editor R. BRADEN : Rfc 1122 : Requirements for internet hosts – communication layers.

[rfc] Rfc 793 : Transmission control protocol, (tcp).

[RFD06] Anirudh RAMACHANDRAN, Nick FEAMSTER et David DAGON : Revealing botnet membership using dnsbl counter-intelligence. *Proc. 2nd USENIX Steps to Reducing Unwanted Traffic on the Internet*, pages 49–54, 2006.

[RL] J Kinsella RSNAKE et RE LEE : Slowloris http dos, juin 2009.

[Roy08] P. ROYAL : Analysis of the kraken botnet, 2008.

[RZPD14] Ashkan RAHIMIAN, Raha ZIARATI, Stere PREDA et Mourad DEBBABI : On the reverse engineering of the citadel botnet. *In Foundations and Practice of Security*, pages 408–425. Springer, 2014.

[SBBD10] Prosenjit SINHA, Amine BOUKHTOUTA, Victor Heber BELARDE et Mourad DEBBABI : Insights from the analysis of the mariposa botnet. *In Risks and Security of Internet and Systems (CRiSIS), 2010 Fifth International Conference on*, pages 1–9. IEEE, 2010.

[SBD+91] Steven R SNAPP, James BRENTANO, Gihan V DIAS, Terrance L GOAN, L Todd HEBERLEIN, Che-Lin HO, Karl N LEVITT, Biswanath MUKHERJEE, Stephen E SMAHA, Tim GRANCE *et al.* : Dids (distributed intrusion detection system)-motivation, architecture, and an early prototype. *In Proceedings of the 14th national computer security conference*, pages 167–176. Citeseer, 1991.

[SEB12] Aditya K SOOD, Richard J ENBODY et Rohit BANSAL : Dissecting spyeye-understanding the design of third generation botnets. *Computer Networks*, 2012.

[Sen] D SENIE : Rfc 2644 : Changing the default for directed broadcast in routers. *Request for Comments*, 2644.

[SGCC+09] B. STONE-GROSS, M. COVA, L. CAVALLARO, B. GILBERT, M. SZYDLOWSKI, R. KEMMERER, C. KRUEGEL et G. VIGNA : Your botnet is my botnet : analysis of a botnet takeover. *In Proceedings of the 16th ACM conference on Computer and communications security*, pages 635–647. ACM, 2009.

[SGHSV11] Brett STONE-GROSS, Thorsten HOLZ, Gianluca STRINGHINI et Giovanni VIGNA : The underground economy of spam : A botmaster's perspective of coordinating large-scale spam campaigns. *In USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2011.

[sha] Shadow server fondation. `https://www.shadowserver.org/`.

[Shi10] A. SHIPP : Lessons in botnets : The after-effects of isp takedowns. RSA 2010, 2010.

[SJ11] Kevin STEVENS et Don JACKSON : Zeus banking trojan report. Rapport technique, mars 2011.

[SLWL08] W Timothy STRAYER, David LAPSELY, Robert WALSH et Carl LIVADAS : Botnet detection based on network behavior. *In Botnet Detection*, pages 1–24. Springer, 2008.

[SM09] Hendrik SCHULZE et Klaus MOCHALSKI : Internet study 2008/2009. *IPOQUE Report*, 37:351–362, 2009.

[SNLDS03] Abhishek SINGH, Ola NORDSTRÖM, Chenghuai LU et Andre LM DOS SANTOS : Malicious icmp tunneling : Defense against the vulnerability. *In Information Security and Privacy*, pages 226–236. Springer, 2003.

[SP01] Dawn Xiaodong SONG et Adrian PERRIG : Advanced and authenticated marking schemes for ip traceback. *In INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 878–886. IEEE, 2001.

[Spa] IP SPAMHAUS : Blocklist.

[Spi03] Lance SPITZNER : Honeypots : Definitions and value of honeypots. *URL www. tracking-hackers. com/papers/honeypots. html (accessed January 3, 2014)*, 2003.

[SPS+01] Alex C SNOEREN, Craig PARTRIDGE, Luis A SANCHEZ, Christine E JONES, Fabrice TCHAKOUNTIO, Stephen T KENT et W Timothy STRAYER : Hash-based ip traceback. *In ACM SIGCOMM Computer Communication Review*, volume 31, pages 3–14. ACM, 2001.

[SSGA+11] Douglas G STEIGERWALD, Brett STONE-GROSS, Ryan ABMAN, Richard KEMMERER, Christopher KRUEGEL et Giovanni VIGNA : The underground economy of fake antivirus software. *In Proceedings of the Workshop on Information Security*, 2011.

[SSPS12] Sérgio SC SILVA, Rodrigo MP SILVA, Raquel CG PINTO et Ronaldo M SALLES : Botnets : A survey. *Computer Networks*, 2012.

[SWKA00] Stefan SAVAGE, David WETHERALL, Anna KARLIN et Tom ANDERSON : Practical network support for ip traceback. *ACM SIGCOMM Computer Communication Review*, 30(4):295–306, 2000.

[SWLL06] W Timothy STRAYER, Robert WALSH, Carl LIVADAS et David LAPSLEY : Detecting botnets with tight command and control. *In Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 195–202. IEEE, 2006.

[TWSO10] Brian K TANNER, Gary WARNER, H STERN et S OLECHOWSKI : Koobface : The evolution of the social botnet. *In eCrime Researchers Summit (eCrime), 2010*, pages 1–10. IEEE, 2010.

[UC13] US-CERT : Dns amplification attacks - alert (ta13- 088a), mars 2013.

[Vir13] VIRUSTOTAL : A propos. *URL https ://www.virustotal.com/fr/about/about/ (accessed January 3, 2014)*, 2013.

[VK99] Giovanni VIGNA et Richard A KEMMERER : Netstat : A network-based intrusion detection system. *Journal of Computer Security*, 7(1):37–71, 1999.

[wal] Judges restraining order takes botnet c&c system offline. `http://arstechnica.com/information-technology/2010/02/judges-restraining-order-takes-botnet-cc-system-offline/`.

[WBH+09] Peter WURZINGER, Leyla BILGE, Thorsten HOLZ, Jan GOEBEL, Christopher KRUEGEL et Engin KIRDA : Automatically generating models for botnet detection. *In Computer Security–ESORICS 2009*, pages 232–249. Springer, 2009.

[WG09] Hailong WANG et Zhenghu GONG : Collaboration-based botnet detection architecture. *In Intelligent Computation Technology and Automation, 2009.*

ICICTA'09. Second International Conference on, volume 2, pages 375–378. IEEE, 2009.

[WS04] Bao-Tung WANG et Henning SCHULZRINNE : A denial-of-service-resistant ip traceback approach. *In Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, volume 1, pages 351–356. IEEE, 2004.

[YPS05] Abraham YAAR, Adrian PERRIG et Dawn SONG : Fit : fast internet traceback. *In INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1395–1406. IEEE, 2005.

[YR12] Sandeep YADAV et A.L.Narasimha REDDY : Winning with dns failures : Strategies for faster botnet detection. *In Security and Privacy in Communication Networks*, volume 96 de *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 446–459. Springer Berlin Heidelberg, 2012.

[YRRR10] S. YADAV, A.K.K. REDDY, AL REDDY et S. RANJAN : Detecting algorithmically generated malicious domain names. *In Proceedings of the 10th annual conference on Internet measurement*, pages 48–61. ACM, 2010.

[YZA08] Wei YAN, Zheng ZHANG et Nirwan ANSARI : Revealing packed malware. *Security & Privacy, IEEE*, 6(5):65–69, 2008.

[zeua] Hackers lock zeus crimeware kit with windows like anti piracy tech. `http://www.computerworld.com/s/article/9170978/Hackers_lock_Zeus_crimeware_kit_with_Windows_like_anti_piracy_tech`.

[Zeub] Zeus gets more sophisticated using p2p techniques. `http://www.abuse.ch/?p=3499`.

[zeuc] Zeus source code leaked. `http://threatpost.com/zeus-source-code-leaked-051011/75217`.

[zeud] Zeus source code repository. `https://github.com/Visgean/Zeus`.

[ZLK10] Chenfeng Vincent ZHOU, Christopher LECKIE et Shanika KARUNASEKERA : A survey of coordinated attacks and collaborative intrusion detection. *Computers & Security*, 29(1):124–140, 2010.

[ZLP+11] Junjie ZHANG, Xiapu LUO, Roberto PERDISCI, Guofei GU, Wenke LEE et Nick FEAMSTER : Boosting the scalability of botnet detection using adaptive traffic sampling. *In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 124–134. ACM, 2011.

[ZTKM11] Xiaonan ZANG, Athichart TANGPONG, George KESIDIS et David J MILLER : Botnet detection through fine flow classification. *unpublished, Report No. CSE11-001*, 2011.