Botnets IRC et P2P pour une supervision à large échelle

Jérôme François, Radu State, Olivier Festor

Botnets IRC et P2P pour une supervision à large échelle

Jérôme François* — **Radu State**** — **Olivier Festor****

* *MADYNES - INRIA Nancy Grand Est, CNRS, Nancy-Université*

** *MADYNES - INRIA Nancy Grand Est*

LORIA - Campus Scientifique - BP 239 - 54506 Vandœuvre-lès-Nancy Cedex, France

{jerome.francois,radu.state,olivier.festor}@loria.fr

RÉSUMÉ. *Alors que le nombre d'équipements à superviser ne cesse de croître, le passage à l'échelle de la supervision des réseaux et services est un véritable enjeu. Un tel challenge semble avoir été par le passé surmonté par les botnets connus actuellement pour être une des principales menaces sur internet car un attaquant peut contrôler des milliers de machines. D'un point de vue technique, il serait très utile de les utiliser dans le cadre de la supervision des réseaux. Cet article propose une nouvelle solution de supervision basée sur les botnets et évalue les performances associées de manière à établir un comparatif détaillé des différents types de botnets utilisables pour la supervision.*

ABSTRACT. *As there are more and more devices to be managed, the scalability of the management of network and services is a real challenge. It seems that attackers have already found a solution in the past by using the botnets which allow the parallel control several thousands of machines on internet. Therefore, a botnet based management solution can be scalable. This article aims to propose such a solution and evaluates the performances of the different types of botnets to compare them for a management solution*

MOTS-CLÉS : *supervision, botnet, IRC, Chord, Slapper, pair-à-pair, configuration à grande échelle*

KEYWORDS: *network management, botnet, IRC, Chord, Slapper, peer-to-peer, mass configuration*

RSTI - TSI – 28/2009. Supervision et sécurité en réseaux, pages 433 à 458

1. Introduction

Plusieurs domaines composent la supervision des réseaux : gestion de défaut, gestion de configuration, gestion comptable, gestion d'exécution et gestion de sécurité. Cependant, un challenge majeur auquel sont confrontées les solutions de supervision est le passage à l'échelle. Une supervision optimale devrait pouvoir se faire à travers internet malgré les nombreux équipements qui pourraient bloquer le trafic légitime de supervision tels que les pare-feux ou encore les routeurs NAT (*Network Addresses Translator*). Les personnes malveillantes ont déjà surmonté le problème en créant des vers ou en utilisant des botnets (réseaux de machines infectées dites bots). On sait par exemple qu'une personne a déjà réussi à contrôler 400 000 machines grâce à un botnet. Il est alors clair qu'adapter un tel mécanisme pour la supervision des réseaux est une solution envisageable à large échelle. Nous allons donc présenter dans cet article le modèle analytique d'une architecture de supervision basée sur les botnets. Notre objectif est de proposer un modèle permettant de valider ou non la viabilité d'une telle solution.

Plus précisement, on cherche à pouvoir faire de la configuration de masse avec les propriétés suivantes :

– passage à l'échelle : un grand nombre de machines à configurer,

– efficacité : nouvelle configuration déployée rapidement,

– anonymat : un attaquant ne doit pas pouvoir découvrir les machines supervisées ainsi que celles qui contrôlent le système pour éviter un détournement de l'usage du botnet,

– sécurité : une machine supervisée ne doit pas pouvoir lancer de fausse requête de supervision sur le réseau,

– garantie d'atteignabilité : l'administrateur doit pouvoir savoir quel est le potentiel résultat de l'envoi d'une requête et notamment le nombre de machines qui l'auront correctement reçu. Cela permet ensuite de prévoir des mesures supplémentaires si ce nombre n'est pas satisfaisant.

La section 2 introduit un exemple d'application pour laquelle notre solution doit être adaptée. Le cadre général de celle-ci est décrite dans la section 3 avant de faire un état de l'art en section 4. La section suivante détaille les différents modèles et métriques qui sont ensuite évalués en section 6. Enfin, nous concluons et présentons les pistes pour nos futures recherches.

2. Exemple d'applications

Bien que de manière générale le but soit de proposer une solution générale de supervision à large échelle, on s'appuie sur un cas de figure permettant d'illustrer l'intérêt d'une solution basée sur les botnets. Une application permettant de traquer les cyber-criminels sur les réseaux pair-à-pair d'échange de fichiers a été proposée dans (Badonnel *et al.*, 2007). Pour cela, des mots-clés spécifiques sont annoncés sur

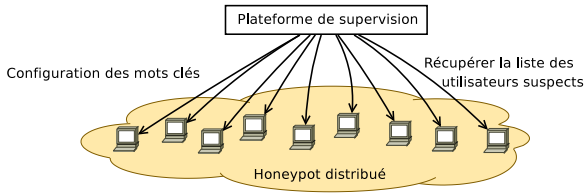


Figure 1. *Cas de figure du honeypot distribué*

le réseau et il suffit alors de récupérer les personnes qui cherchent ces mots-clés. De manière à améliorer la détection de ces personnes, il faudrait qu'un grand nombre de pairs annoncent ces mots-clés. On propose donc une architecture de type honeypot distribuée sur la figure 1 où tout utilisateur du réseau qui le souhaite peut participer. Un honeypot est une machine qui attire un attaquant en se faisant passer pour une machine vulnérable comme dans (Provos, 2004) ou comme dans le cas présent une machine qui propose du contenu spécifique. La tâche principale de supervision est de pouvoir configurer les honeypots en leur fournissant les différents mots-clés ce qui correspond bien à faire de la configuration en masse.

On retrouve alors les propriétés énoncées précédemment et de manière plus précise on souhaite avoir :

– passage à l'échelle : tout utilisateur doit pouvoir participer, ce qui peut donc correspondre à des centaines de milliers de machines,

– efficacité : les mots-clés sont spécifiques mais pas forcément en lien avec le contenu et peuvent donc changer assez rapidement,

– anonymat : un attaquant ne doit pas pouvoir attaquer une partie du système pour se protéger lors d'éventuels téléchargements,

– sécurité : un attaquant ne doit pas pouvoir annoncer une liste de faux mots-clés,

– garantie d'atteignabilité : l'administrateur doit savoir quel est le pourcentage de machines qui a bien la liste de nouveaux mots-clés à chaque mise à jour.

3. Applications malveillantes adaptées pour la supervision

Il existe plusieurs types de logiciels malveillants dont les vers. Même si un ver se caractérise d'abord par sa capacité à contaminer des machines non infectées, il est aussi généralement associé à un programme annexe. Il peut par exemple faire de la machine un bot, c'est-à-dire que celle-ci attend les ordres d'un attaquant (lancer une attaque de déni de service, récupération de données personnelles). En effet, les machines infectées exécutent ce programme et se connectent à un même réseau qui véhicule les commandes de l'attaquant ainsi que les réponses. Elles forment ainsi un botnet.

Dans une étude précédente (State *et al.,* 2007), une première ébauche d'une possible application des logiciels malveillants pour la supervision des réseaux était proposée. Dans le cas d'un botnet, les communications sont simples et se basent sur une architecture décentralisée ce qui permet de commander un grand nombre de bots comme 400 000 machines (McLaughlin, 2004). Un des moyens les plus connus est l'utilisation d'un réseau IRC (Oikarinen *et al.*, 1993) dont le but initial est de permettre aux utilisateurs de discuter entre eux. La référence (Schiller *et al.*, 2007) donne de nombreux détails quant aux fonctionnements des botnets. D'autres botnets se basent sur des réseaux pair-à-pair. Les différentes observations faites sur les botnets laissent penser qu'une organisation de la supervision peut être basée sur un botnet. On considère donc que les clients (bots) sont les équipements à superviser et que le superviseur est un client particulier (botmaster). Nous allons considérer plusieurs cas dans ce papier : un botnet IRC, deux botnets pair-à-pair (Slapper (Arce *et al.,* 2003) et Chord (Stoica *et al.*, 2001)). Cela permettra donc d'évaluer différentes solutions possibles avec des degrés de décentralisation différents car un réseau IRC est faiblement décentralisé alors que les réseaux pair-à-pair le sont totalement. Nous avons choisi d'étudier deux réseaux pair-à-pair différents car l'organisation du réseau n'est pas la même dans chacun des cas. Notre objectif est d'élaborer un modèle analytique qui permettra d'évaluer les performances d'un botnet et aussi de permettre à un administrateur d'évaluer ce que peut lui apporter une telle solution. Nous ne considérons pas la phase de déploiement du botnet mais seulement son utilisation.

4. Etat de l'art

Les botnets permettent aux attaquants de contrôler facilement de très nombreuses machines en même temps. Un réseau IRC est le plus souvent utilisé comme canal de contrôle (Barford *et al.*, 2006). Cependant de nouvelles formes apparaissent comme celles basées sur les réseaux pair-à-pair (Grizzard *et al.*, 2007). Dans (Wang *et al.*, 2007), une analyse de plusieurs botnets pair-à-pair est faite et les auteurs en proposent même un nouveau type. Nous proposons dans ce papier des modèles permettant d'étudier les performances des botnets. Concernant la supervision, (Ohno *et al.*, 1995) avaient aussi proposé d'utiliser un mécanisme malveillant pour faire de la supervision. En fait, un ver patrouille sur différentes machines avant de retourner vers la plateforme de management. Cependant les études et évaluations restent très limitées. Le vers Code-Green (Szor, 2005) est aussi très connu car c'est un ver qui désinfecte les machines infectées par Code-Red. Le problème de ce type de vers est la légalité car la propagation reste souvent incontrôlée. Dans une solution classique de supervision, le superviseur communique directement avec chaque équipement à superviser. Comme l'architecture centralisée limite le passage à l'échelle, plusieurs solutions décentralisées sont apparues. On peut par exemple citer la supervision par délégation (Goldszmidt *et al.*, 1995) ou encore l'utilisation de niveaux hiérarchiques et de superviseurs en cascade (SNMP *et al.*, 2007). En fait, chaque superviseur intermédiaire doit transformer les requêtes de supervision de manière à ce que celles-ci soient adaptées aux différents équipements dont ils sont responsables. Les réponses doivent aussi être

transformées ou agrégées comme dans (Lim *et al.*, 2005). Dans ces cas, les requêtes pour les différents équipements ne sont pas identiques. Dans ce papier le contexte est celui de la configuration de masse, c'est-à-dire que la même requête doit être envoyée à tous les équipements. Dans ce cas, il n'y a plus besoin d'avoir des superviseurs intermédiaires qui peuvent être soumis à des pannes ou à des attaques supplémentaires. Une extension de la supervision hiérarchique basée sur les réseaux pair-à-pair est proposée dans (Artigas *et al.*, 2006). (Ahmed *et al.*, 2007) proposent de résoudre des recherches dans les réseaux pair-à-pair avec des caractères génériques en se basant sur des techniques d'agrégation et de réplication. *Echo Pattern* (Lim *et al.*, 2001) est un modèle proche des réseaux pair-pair qui promet de bonnes performances. Il existe aussi des solutions de supervision utilisant les réseaux actifs (Schwartz *et al.*, 2000). Dans ce cas une requête est un programme léger transmis entre les différents hôtes du réseau. Cependant, leur déploiement est très limité car il nécessite des équipements dédiés. Certaines solutions sont dédiées aux réseaux Ad-Hoc comme (Badonnel *et al.*, 2006) où le réseau est découpé en groupes avec pour chacun d'eux un responsable élu. Pour un botnet, aucune machine spécifique n'a besoin d'être utilisée hormis la plateforme de management car tout le monde peut participer et les ressources nécessaires sont faibles contrairement à la plupart des autres solutions.

5. Modèles

5.1. *Paramètres*

Cette section décrit les paramètres communs aux différents modèles :

– m : le facteur de branchement maximal représente le nombre maximal de voisins d'un nœud,

– N : le nombre total de nœuds dans le réseau,

– β : la probabilité pour un nœud d'être compromis. On considère qu'un attaquant qui a compromis un nœud est seulement capable d'écouter l'ensemble des communications de ce nœud et donc de découvrir les autres nœuds,

– $\alpha(m)$ est le facteur de disponibilité. Un nœud disponible peut faire suivre les requêtes de supervision. Ce facteur modélise le fait qu'un nœud n'a plus assez de ressources. On considère que ce facteur est une fonction décroissante de m car plus un nœud doit maintenir de connexions plus il risque d'être surchargé.

Nos modèles devront permettre de déterminer le nombre de nœuds et donc d'équipements que l'on peut atteindre lors de l'envoi d'une requête de supervision. Pour cela, nous définirons l'atteignabilité, une métrique permettant d'évaluer l'efficacité de l'envoi d'un message broadcast. Il est à noter que nous considérons que nous sommes à un instant précis et que β représente aussi le pourcentage moyen de nœuds compromis que l'on pourrait observer. Ainsi β est une probabilité moyenne. Pour évaluer plus précisément l'impact d'une attaque, il faudra dans nos travaux futurs prendre en

compte le fait que la probabilité qu'un nœud soit compromis est différente d'un nœud à l'autre car il y a plus de chances d'être compromis si le nœud voisin l'est.

5.2. *Modèle d'un botnet IRC*

Un réseau IRC est composé de plusieurs serveurs interconnectés ensemble de manière à former un arbre de diffusion. A chaque fois qu'un utilisateur envoie un message, celui-ci est relayé à tous les serveurs et ainsi à tous les utilisateurs. De manière plus précise, les utilisateurs sont regroupés dans différents canaux équivalents conceptuellement à un groupe multicast. Dans un botnet IRC, le contrôleur (Botmaster) envoie les commandes au serveur auquel il est connecté et qui est responsable de la diffusion *via* l'arbre de diffusion. Pour l'instant nous ne considérerons que les serveurs que l'on nommera aussi nœuds. L'objectif est donc de calculer le nombre moyen de serveurs qui sont atteints à une distance donnée (sauts applicatifs). On commence donc par calculer le nombre moyen de voisins directs puis on généralise la formule.

Chaque serveur a un nombre maximum m de voisins : c'est le facteur de branchement maximal. Le nombre de connexions qu'un serveur maintient avec d'autres est donc compris entre 1 et m. La probabilité p_k est la probabilité pour un nœud d'avoir k serveurs comme voisins. Plusieurs cas sont à prendre en compte. Tout d'abord, p_0 signifie que le serveur est déconnecté (hors-service). On obtient donc $p_0 = 1 - \alpha(m)$. Ensuite, il est impossible d'avoir $k > m$ d'où $p_k = 0$ pour $k > m$. Enfin nous considérons que pour les autres cas "normaux" la distribution est aléatoire et uniforme donc $p_k = \alpha(m) \times \frac{1}{m} = \dfrac{\alpha(m)}{m}$ pour $1 \leq k \leq m$. Cette distribution reflète que nous considérons qu'il n'y a apparemment aucune topologie spécifique meilleure qu'une autre.

Grâce à cette fonction de distribution nous pouvons facilement calculer la fonction génératrice :

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k = \sum_{k=0}^{m} p_k x^k = p_0 + \sum_{k=1}^{m} p_k x^k = p_0 + \frac{\alpha(m)}{m} \times \sum_{k=1}^{m} x^k$$

Cette fonction permet de calculer directement le nombre moyen de voisins : $\mathbb{E}(k) = G_0'(1)$. Il est alors possible de calculer la fonction génératrice $G^j(x)$ de la fonction de distribution du nombre de nœuds à une distance j d'une manière similaire à (Ramachandran *et al.*, 2006) :

$$G^j(x) = \begin{cases} G_0(x) & \text{pour} \quad j = 1 \\ G^{j-1}(G_1(x)) & \text{pour} \quad j \geq 2 \end{cases} \qquad [1]$$

$G_1(x) = \dfrac{G_0'(x)}{G_0'(1)}$ est la fonction de distribution du nombre de connexions sortantes d'un nœud atteint de degré k.

On peut alors calculer le nombre moyen de nœuds à une distance j du nœud d'origine qui est $z_j = (G^j)'(1)$. Une fois de plus, en utilisant une méthode similaire à (Ramachandran *et al.*, 2006), on obtient les formules suivantes :

$$z_1 = (G^1)'(1) = G'_0(1) \qquad z_2 = (G^1(G_1))'(1) = G''_0(1) \qquad z_j = \left[\frac{z_2}{z_1}\right]^{j-1} z_1$$

On peut alors facilement calculer les différents termes car p_0 est constant :

$$G'_0(x) = \frac{\alpha(m)}{m}\sum_{k=1}^{m} kx^{k-1} \qquad G''_0(x) = \frac{\alpha(m)}{m}\sum_{k=1}^{m} k(k-1)x^{k-2}$$

5.2.1. *Atteignabilité*

L'atteignabilité est la proportion moyenne de serveurs atteints à une distance k donnée, ce qui est un bon indicateur de passage à l'échelle. Le nombre de serveurs atteints est calculé grâce aux formules précédentes. Cependant, un attaquant peut être intéressé pour compromettre le botnet de manière à récupérer des informations sensibles sur le réseau et les équipements de l'entreprise. L'attaquant peut surveiller tous les échanges réseaux d'un nœud avec la probabilité β et donc découvrir les autres serveurs. On peut donc considérer que le réseau reste opérationnel si aucun des nœuds n'est compromis. On multiplie donc la proportion de nœuds atteints par la probabilité d'avoir le réseau non découvert qui est $(1-\beta)^N$ c'est-à-dire qu'aucun des N nœuds ne soit compromis. L'atteignabilité peut alors être complètement définie :

$$att(k) = \frac{(1-\beta)^N \times min(\sum_{j=1}^{k} z_j, N)}{N} \qquad [2]$$

L'atteignabilité des bots/équipements est équivalente. En effet, nous considérons le cas où les bots sont répartis aléatoirement et uniformément sur tous les serveurs. Le nombre de bots ou d'équipements atteints à une distance k est donc $bots(k) = att(k) \times B$. La proportion de bots atteints est alors exprimée de la manière suivante : $att_{bots}(k) = \dfrac{bots(k)}{B} = att(k)$

5.2.2. *Atteignabilité moyenne*

De manière à obtenir un indicateur global du passage à l'échelle, l'atteignabilité moyenne est l'atteignabilité moyenne de toutes les distances possibles sachant que la plus courte des distances est 1 et la plus longue est N lorsque l'arbre n'est en fait qu'une chaîne :

$$avg\_att(k) = \frac{\sum_{k=1}^{N} att(k)}{N} \qquad [3]$$

5.2.3. *Délais*

Le temps nécessaire pour envoyer une requête de supervision est un élément déterminant dans le choix d'une solution de supervision car les contraintes temporelles sont souvent fortes. Pour pouvoir l'évaluer il faut considérer deux paramètres : le temps t_h pour envoyer un message vers un équipement final et le temps t_s pour envoyer un message vers un autre serveur. Dans le cas d'une solution centralisée, la plateforme de supervision va envoyer successivement les messages vers l'ensemble des machines : $temps_{centralise} = C \times t_h$ où C est le nombre de machines. Dans le cas d'un botnet IRC, les serveurs peuvent transmettre les messages aux serveurs avant de les transmettre aux machines. Considérons $C_{serveur}$, le nombre moyen d'équipements connectés à un même serveur, le temps total est donc composé du temps pour atteindre le dernier serveur à une distance k plus le temps pour ce serveur de transmettre les requêtes à l'ensemble des machines connectées :

$$temps_{botnet} = k \times t_s + C_{serveur} \times t_h \quad [4]$$

Grâce à cette métrique, un administrateur peut évaluer le temps de chaque opération et ainsi les ordonner dans le cas où certaines opérations urgentes doivent être terminées en priorité.

5.3. *Modèle d'un botnet Slapper*

Le vers Slapper (Arce *et al.*, 2003) est apparu pour la première fois en 2002. Il ne se limite cependant pas au seul rôle d'un ver puisqu'il est accompagné d'un programme annexe sophistiqué permettant la construction d'un botnet. En effet, chaque machine infectée pouvait être contrôlée à distance par l'attaquant pour envoyer des courriers indésirables par exemple. Le botnet formé était en fait un réseau pair-à-pair de type totalement maillé : chaque machine connaissant toutes les autres machines infectées. L'attaquant peut alors envoyer un message vers l'une d'entre elles spécifiquement par l'intermédiaire des autres. Pour cela, un nombre maximal de sauts est défini. Lorsqu'une machine reçoit le message, si le nombre de sauts est zéro alors le message est directement envoyé à la destination puisque le réseau est entièrement maillé. Sinon plusieurs nœuds choisis au hasard sont sélectionnés pour transmettre le message et dans ce cas, le nombre de sauts est décrémenté. Ce protocole permet alors à l'attaquant de rester caché puisque plusieurs nœuds intermédiaires sont utilisés. Il se préserve ainsi d'être détecté par une sonde car aucune machine ne sait si le nœud qui lui a envoyé le message est l'émetteur initial, c'est-à-dire l'attaquant. Enfin pour permettre une réponse de la part de la destination, chaque machine qui route un message garde en mémoire l'émetteur de ce message ainsi qu'un identifiant contenu dans le message. Cela permet alors de router la réponse car le chemin de retour peut être créé à partir de cette information stockée.

Dans le contexte de la configuration de masse, Slapper est doté d'un mécanisme de broadcast appelé *broadcast segmentation*. Même si le contrôleur ou l'attaquant

étaient capables d'envoyer directement un message à tous les hôtes du réseau formé par Slapper, cela pose deux problèmes :

– passage à l'échelle : initialisation d'une connexion et échange avec chaque machine,

– sécurité : on connaît le contrôleur qui est l'émetteur du message.

Pour pallier ces problèmes, le contrôleur décide d'envoyer le message à deux machines choisies au hasard. Celles-ci répètent la même procédure et ainsi de suite. De manière à arrêter la propagation infinie du message, un nombre de sauts maximal est fixé. Cependant comme les nœuds sont choisis au hasard, il est clair qu'il est possible que certains nœuds ne reçoivent jamais le message. La figure 2 illustre ce mécanisme où les nœuds sont visuellement placés de la même manière que Chord pour faciliter la comparaison. Le contrôleur se situe sur le nœud 0 et il envoie un message à deux pairs aléatoires : 4 et 8. Ces derniers répètent le même processus. Le non déterminisme de Slapper est présenté par l'intermédiaire de deux cas de figures possibles pour ce deuxième saut car les nœuds sont choisis au hasard. Comme chaque nœud en choisit deux autres, on peut considérer que le facteur de branchement de Slapper est 2. Cependant nous allons considérer un cas plus général où celui-ci est paramétrable ce qui correspond à notre paramètre m.

Supposons que t nœuds aient déjà reçu le message sur un total de N nœuds. Ils vont alors transmettre c messages identiques à leur tour à d'autres pairs. On peut alors déterminer la probabilité $p(N,t,c,j)$ d'envoyer ces messages à j pairs qui ne l'ont pas encore reçu :

$$p(N,t,c,j) = \frac{C_{N-t}^{j} \times \Gamma_{t+j}^{c-j}}{\Gamma_{N}^{c}} \qquad [5]$$

avec $\Gamma_n^k = C_{n+k-1}^k$ la combinaison avec répétitions de k éléments parmi n.

Dans cette équation, le premier terme au numérateur est le nombre de possibilités de choisir les j nœuds nouvellement contactés parmi l'ensemble c'est-à-dire N. Parmi l'ensemble des c messages, il y en a donc j qui sont utilisés pour envoyer à ces nœuds précisément. Le second terme représente alors le nombre de destinations possibles des $c-j$ messages restants. Ces derniers peuvent alors être envoyés à n'importe quel pair parmi ceux déjà contactés auparavant ainsi que ceux nouvellement contactés soit $t+j$. Le dénominateur représente quant à lui le nombre total de possibilités pour choisir la destination des c messages parmi l'ensemble des nœuds. Cette probabilité ne peut être calculée que pour un nombre cohérent de pairs c'est dire pour $j \leq N-t$.

Le fonctionnement du broadcast fait qu'au $ième$ saut, le nombre de messages dupliqués reçus est m^i. Cependant nous considérons qu'un nœud peut être en surcharge et on considère que le nombre de messages dupliqués est limité à $msg = (m \times \alpha(m))^i$ car chacun des nœuds a une probabilité $\alpha(m)$ de fonctionner normalement. Ce nombre de messages limite donc aussi le nombre de nouveaux nœuds qui vont recevoir l'infor-

mation. Ce dernier est aussi limité par le nombre de nœuds qui n'ont pas encore reçu de message. On peut donc définir totalement le nombre maximal de nouveaux pairs qui peuvent recevoir le message :

$$max_{msg} = min(msg, N - att(N, m, i-1))$$

où $att(N, m, i-1)$ représente le nombre de nœuds ayant reçu un message au saut précédent $i-1$.

Grâce à la formule [5], il est alors possible de connaître le nombre moyen de nouveaux nœuds atteints à une distance i en considérant les différents cas possibles c'est-à-dire de 0 à max_{msg} :

$$r(N, m, i) = \sum_{k=0}^{max_{msg}} p(N, att(N, m, i-1), msg, k) \times k$$

Pour le cas $i = 0$, on considère seulement le nœud d'origine. Dans les autres cas, le nombre moyen de nœuds atteints à une distance i est composé des nouveaux nœuds atteints ainsi que ceux ayant déjà reçu le message. On a donc la formule d'atteignabilité suivante :

$$att(N, m, i) = \begin{cases} 1 & \text{si } i = 0 \\ reach(N, m, i-1) + r(N, m, i) & \text{sinon} \end{cases} \quad [6]$$

Enfin, il est nécessaire de prendre en compte le fait que si un nœud est compromis, alors l'attaquant peut découvrir l'ensemble du réseau en écoutant les communications car le réseau est totalement maillé. Ainsi l'atteignabilité est multipliée par $(1-\beta)^N$ de manière similaire au modèle IRC.

5.4. *Modélisation de Chord*

Un réseau totalement maillé tel que Slapper le construit présente des limites en termes de passage à l'échelle car chacun doit avoir connaissance de tous. Une évolution en cours des logiciels malveillants est l'utilisation des réseaux pair-à-pair structurés. Nous avons choisi d'étudier Chord, un des plus connus, qui a montré de bonnes performances dans de nombreuses études (Stoica *et al.*, 2001) (Dabek *et al.*, 2001). Sur un réseau Chord, chaque nœud ou pair participant a un identifiant qui lui est associé tel que $0 \leq id < N_{MAX}$ où $N_{MAX} = 2^k, k \in \mathbb{I}^{+*}$. N_{MAX} est le nombre maximal de nœuds participants ainsi que la taille de l'espace des identifiants. On suppose que les identifiants des nœuds soient aléatoirement et uniformément répartis dans cet espace et que la distance moyenne entre deux identifiants consécutifs est d. On a donc $N_{MAX} = N \times d$. Contrairement à un nœud Slapper qui connaît l'ensemble des autres nœuds, un nœud Chord n'a qu'une vue partielle. La taille de la table de routage

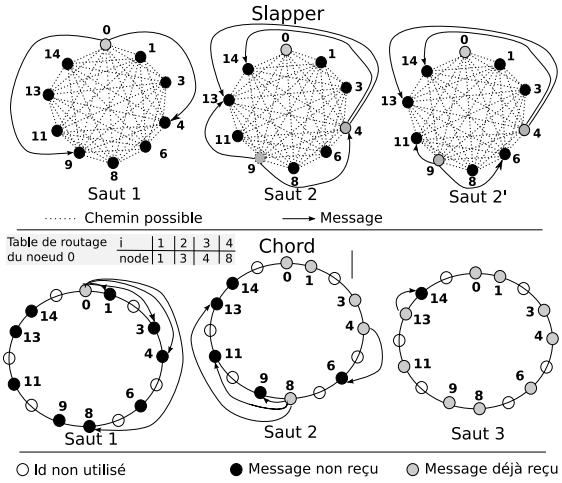


Figure 2. *Brodcast dans les réseaux P2P*

d'un nœud est de $log(N_{MAX})$. La $ième$ entrée du nœud d'identifiant id_1 contient le nœud dont l'identifiant id_2 est le plus petit possible tel que $id_1 + 2^{i-1} \leq id_2$.

Comme la distribution des identifiants est uniforme, on peut considérer que les premières entrées sont identiques car le successeur est toujours le même. En moyenne, la première entrée i qui est différente est celle telle que $2^{(i-1)} > d$. Le nombre total d'entrées différentes est donc :

$$log(N_{MAX}) - log(d) = log(\frac{N_{MAX}}{d}) = log(N) \quad [7]$$

On calcule en fait la différence entre le nombre maximal d'entrées différentes et la première entrée différente à laquelle on rajoute 1 ce qui correspond à la première entrée de la table de routage. Les nœuds Chord sont représentés selon le sens des aiguilles d'une montre sur la figure 2. La $ieme$ entrée de la table de routage du nœud p est l'identifiant du premier nœud à une distance d'au moins 2^{i-1}. (Stoica *et al.*, 2001) donnent tde plus amples détails à propos du fonctionnement de Chord et notamment sur la mise à jour de la table de routage. Les auteurs de (El-Ansary *et al.*, 2003) ont proposé un mécanisme de broadcast efficace grâce auquel un nœud de Chord ne reçoit qu'une seule fois le message. Nous avons choisi de nous baser sur cet algorithme pour diffuser une requête au sein du botnet. Bien que d'autres réseaux pair-à-pair structurés existent, notre choix s'est porté sur Chord car même si chacun d'eux possèdent leurs particularités, tous sont basés sur une table de hachage distribuée. Dans le cas du broadcast, cela n'amène pas à beaucoup de différences car dans la suite. On ne rentre pas ici dans des considérations de déploiement, de dynamique et de mise à jour du réseau. Pour en revenir au broadcast, un message contient une limite d'exploration. Lorsque le contrôleur envoie le message, il l'envoie à chaque entrée différente dans la

table de routage et chacun avec une limite d'exploration différente qui est en fait l'entrée suivante dans la table de routage (ou le nœud contrôleur pour la dernière entrée). Chaque nœud répète ce processus sauf qu'un nœud p n'enverra jamais un message au nœud dont l'identifiant est supérieur à la limite d'exploration. De même, le dernier nœud qui vérifie cette condition se verra attribuer une limite d'exploration identique à celle du nœud p. Ce mécanisme est illustré sur la figure 2. Le contrôleur est le nœud 0 et envoie le message aux différentes entrées de sa table de routage 1, 3, 4 et 8 avec respectivement comme limite d'exploration 3, 4, 8, 0. Il est important de souligner que cette limite est exclusive et c'est pour cela qu'à la deuxième itération (deuxième saut), le nœud 4 n'envoie que le message au nœud 6 car sa limite est 8. La table de routage d'un nœud contient au plus $log(N_{MAX})$ entrées. Chaque nœud transmettra donc le message broadcast à $log(N_{MAX})$ autres nœuds tout au plus. On peut donc en déduire que le facteur de branchement est $m = log(N_{MAX})$ avec normalement N_{MAX} qui est une puissance de 2.

A partir de ces différentes hypothèses et conclusions, l'algorithme 1 calcule pour un identifiant donné, le nombre de sauts nécessaires pour atteindre le nœud correspondant à partir du nœud 0. Cet algorithme ne peut être appliqué que pour les identifiants valides c'est-à-dire que $\frac{id}{d}$ doit être une valeur entière et positive. Si cet identifiant est une puissance de 2 alors le nœud peut être atteint directement. Sinon, s'il n'y a aucun identifiant valide entre l'identifiant du nœud à contacter et la puissance de 2 précédente inclus, on peut aussi contacter le nœud en un seul saut. Dans tous les autres cas, il existe un nœud intermédiaire. On calcule alors le nombre de sauts pour ce nœud intermédiaire et on obtient le nombre de sauts total de manière récursive. Comme on peut le constater nous avons considéré que le contrôleur était le nœud 0. Cependant il n'est pas nécessaire de considérer les autres cas qui sont équivalents à décaler l'ensemble des identifiants des nœuds.

Algorithme 1 Nombre de sauts

```
1: id = d × k l'identifiant du nœud
2: entier(x) retourne la partie entiere de x
3: l = log(id)
4: si l ∈ Integer alors
5:     retourner 1
6: sinon
7:     total ← 0
8:     count ← 1
9:     tantque l ≠ entier(l) faire
10:        total ← total + 2^entier(l)
11:        si id − total < d alors
12:            retourner count
13:        sinon
14:            l ← log(id − total)
15:        finsi
16:    fin tantque
17: finsi
```

On peut alors exécuter cet algorithme pour chaque identifiant. L'atteignabilité au $ieme$ saut avec N nœuds au total est le nombre de nœuds à une distance i du nœud d'origine : le nœud contrôleur.

De manière à pouvoir comparer les différents modèles, il est nécessaire d'intégrer les différents types de pannes. Dans le pire des cas, il y a une panne qui empêche l'envoi du message lors du premier saut. Dans ce cas, un autre nœud avec un identifiant plus petit est choisi pour le remplacer. Il est alors capable de contacter le nœud qui n'avait pas pu être contacté car la distance entre les 2 est plus petite. On réitère ce processus en considérant que si un nœud est à une distance h alors il y a au plus h pannes. Pour chacune d'entre elles, un saut supplémentaire sera nécessaire dans le pire cas. La probabilité $p(f < h)$ d'avoir $f < h$ pannes est définie par une loi binomiale. On peut alors calculer cette probabilité puis la multiplier par le nombre de nœuds atteints à la distance h de manière à rajouter ce résultat au nombre de nœuds atteints à une distance $h + f$. L'algorithme 2 fait donc ces différents calculs par ordre croissant des distances h.

Algorithme 2 Effets des pannes

h le nombre de sauts
n le nombre de nœuds à la distance h
$total \leftarrow 0$
MAX_HOP est la distance maximale
$reach[x]$ est le nombre moyens de nœuds atteints à la distance x
pour $i \leftarrow 1$ toMAX_HOP **faire**
    pour $i \leftarrow 1$ to h **faire**
        $p \leftarrow n \times *C_h^i * \alpha(m)^{k-i} * (1 - \alpha(m))^i$
        $reach[h + i] \leftarrow reach[h + i] + p$
        $total = total + p$
    fin pour
fin pour
$reach[h] \leftarrow reach[h] - total$

Contrairement à Slapper et à un réseau IRC, un nœud compromis par un attaquant n'implique pas que l'ensemble du réseau soit inutilisable. Nous ne considérons pas ici la possibilité pour l'attaquant de contrôler le nœud et donc d'injecter des requêtes (problèmes pouvant être réglés en ajoutant une couche de cryptographie) mais seulement la possibilité d'écouter les communications. Dans le pire des cas, un attaquant peut découvrir tout le réseau s'il a compromis $n\_compromis = \frac{N}{log(N)}$ nœuds car chacun des nœuds connaît $log(N)$ autres nœuds. Bien entendu, nous considérons ici le pire des cas où chacune d'elles est différente. En supposant que la probabilité qu'un nœud soit compromis est β, on peut utiliser une loi binomiale pour calculer la probabilité que le réseau soit entièrement vulnérable c'est-à-dire qu'il y ait $n\_compromis$ nœuds compromis. Il est alors possible d'approximer cette loi par une loi de Poisson avec $\lambda = N \times \beta$. La probabilité que le réseau soit entièrement découvert est donc :

$$e^{-\lambda} \sum_{i=n\_compromis}^{N} \frac{\lambda^i}{i!}$$

Les différentes valeurs d'atteignabilité sont alors multipliées par la probabilité inverse qui permet de prendre en compte la robustesse du système si un attaquant essaie de détourner l'usage du réseau à son propre compte.

5.5. *Délai dans les réseaux P2P*

Considérons le temps t_h pour envoyer un message d'un nœud à l'autre. Considérons une distance de k sauts. Le temps nécessaire pour transmettre le message aux nœuds du premier saut est m. On peut alors répéter ce processus pour chaque saut. Cependant on peut considérer que dès qu'un nœud reçoit un message alors il commence tout de suite à le retransmettre également. Pour calculer le délai total de la propagation d'une requête on considère donc le temps nécessaire pour atteindre le dernier nœud à chaque saut. Au premier saut il faudra attendre $m \times t_h$ ainsi qu'au deuxième et ainsi de suite. Au bout de k sauts le délai total est donc :

$$m \times k \times t_h \qquad [8]$$

6. Résultats

L'objectif de cette section est de déterminer les performances d'une architecture de management basée sur les botnets selon différentes configurations. On cherche notamment à savoir si un tel système peut passer à l'échelle selon différentes topologies (différents m). On cherchera aussi à savoir combien il faut de serveurs pour avoir de bonnes performances. Nous avons utilisé l'outil Maxima (Maxima, 2007) permettant de faire des calculs formels et plus précisement de calculer des séries.

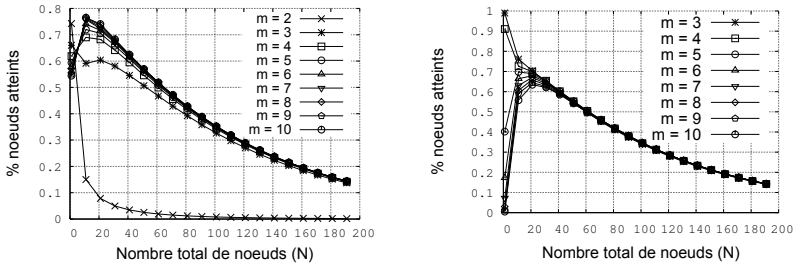
6.1. *Modèle IRC*

6.1.1. *Paramètres*

$\alpha(m)$ est une fonction décroissante et doit être comprise entre 0 et 1 pour m allant de 2 à l'infini car on exclut le cas $m = 1$ correspondant à une topologie de deux nœuds seulement. Plusieurs possibilités peuvent être utilisées :

- $\alpha(m) = \alpha_1(m) = 1/m$
- $\alpha(m) = \alpha_2(m) = e^{(m-i)}$

La deuxième fonction décroît moins rapidement mais dépend d'un paramètre i qui sera choisi de manière à avoir des valeurs assez élevées lorsque le facteur de branchement m est petit. Par exemple, si on fait des simulations avec m variant de 3 à 5, on fixera $i = 3$. Un tel α revient donc à un environnement instable où il est difficile de maintenir actives de nombreuses connexions. La probabilité pour un nœud d'être découvert est $\beta = 0.01$. Ces paramètres ont été fixés arbitrairement mais représentent quand même le système dans un environnement hostile car, par exemple, pour seulement 20 serveurs IRC, β induit une probabilité d'être totalement découvert de 20 %. Ces paramètres doivent être en réalité adaptés aux contraintes spécifiques à l'environnement où est déployé le système. Dans le cas présent, nous avons fait en sorte de nous placer dans une situation défavorable pour le botnet.



(a) $\alpha(m) = \alpha_1(m)$ (b) $\alpha(m) = \alpha_2(m)$

Figure 3. *Atteignabilité moyenne selon différents facteurs de branchement*

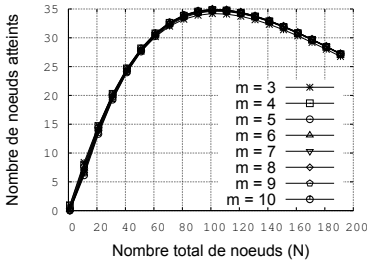
6.1.2. *Atteignabilité moyenne*

Dans un premier temps, l'atteignabilité moyenne sera évaluée selon différents facteurs de branchement et le nombre total de serveurs/nœuds dans le réseau. Sur la figure 3(a), m varie de 1 à 10 et $\alpha_1(m)$ est utilisé. Au début, comme il y a de plus en plus de nœuds dans le réseau, plus de nœuds peuvent être atteints et la courbe est donc croissante. Cependant, à partir d'un certain moment proche de $N = 10$, cela ne se vérifie plus car β limite le nombre de nœuds atteints. On en déduit qu'en terme d'atteignabilité, déployer un réseau de plus de 10 nœuds a des performances limitées. Le cas $m = 2$ peut être considéré comme inutilisable. En fait, dans ce cas précis, l'arbre n'est qu'une chaîne car chaque nœud possède un lien vers son parent et un autre vers son enfant.

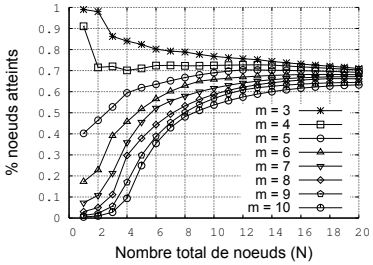
Si l'on considère maintenant $\alpha(m) = \alpha_2(m)$ sur la figure 3(b). La différence se situe au début car les courbes représentant un petit facteur de branchement sont moins affectées, c'est-à-dire $\alpha_2(m) > \alpha_1(m)$. Ainsi on peut en déduire que la première partie de la courbe dépend de $\alpha(m)$ alors que la seconde partie, ici après 20 nœuds, dépend de β.

Considérons maintenant le nombre de nœuds atteints et non plus la proportion correspondante. On remarque sur la figure 4(a) que toutes les courbes sont similaires et qu'elles atteignent une valeur maximale de 35 nœuds pour 100 nœuds au total. Il n'y a donc aucune raison d'avoir un réseau de plus de 100 serveurs. Un administrateur pourra alors choisir la topologie du réseau et ce qu'il veut maximiser (le nombre de serveurs atteints ou l'atteignabilité) selon les contraintes techniques et financières imposées. Par exemple, avec 100 bots connectés à un serveur, il est possible dans le meilleur des cas de superviser 3500 machines.

Ce résultat peut être amélioré en diminuant le facteur de branchement. Hormis pour $m = 2$, les courbes sont différentes pour des petites valeurs de N. Avec $\alpha_1(m)$, plus le facteur de branchement est élevé plus l'atteignabilité est élevée contrairement au cas $\alpha_2(m)$ présenté sur la figure 4(b). L'atteignabilité dépend de deux forces an-

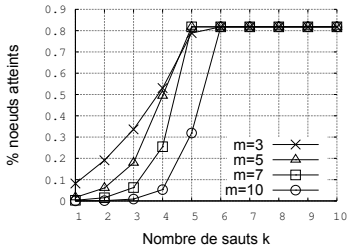


(a) Nombre absolu de nœuds atteints

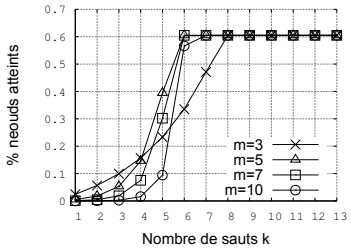


(b) Atteignabilité moyenne avec $1 \leq N \leq 20$

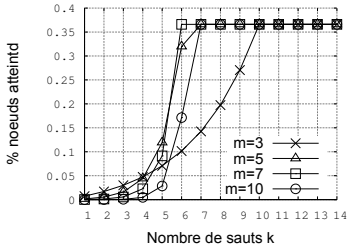
Figure 4. *Nœuds atteints avec* $\alpha(m) = \alpha_2(m)$



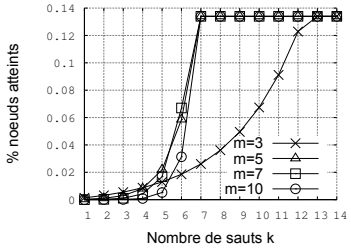
(a) $N = 20$



(b) $N = 50$



(c) $N = 100$



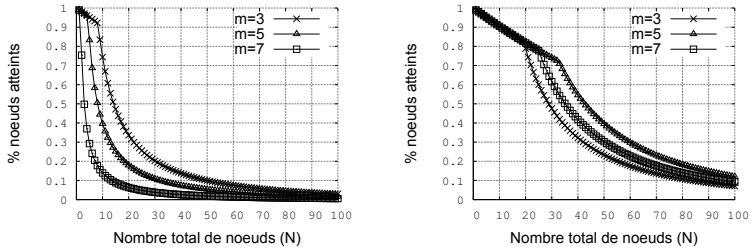
(d) $N = 200$

Figure 5. *Dépendance entre atteignabilité et facteur de branchement*

tagonistes : le facteur de branchement et la probabilité d'avoir une machine en panne c'est-à-dire $1 - \alpha(m)$.

6.1.3. *Nombre de sauts*

Nous considérons maintenant différents N (nombre total de serveurs) respectivement 20, 50, 100 et 200 de manière à voir l'évolution de l'atteignabilité selon le nombre de sauts sur, respectivement, les figures 5(a), 5(b), 5(c) et 5(d). On



(a) j = 3 (b) j = 5

Figure 6. *Atteignabilité selon N et le nombre de sauts*

constate que toutes les courbes sont limitées par un certain plafond qui dépend de β à cause de la formule (2). Par exemple, pour $N = 20$, le plafond est égal à $(1 - \beta)^{20} = (1 - 0.01)^{20} = 0.81$ qui est la probabilité de ne pas être découvert. Connaître le nombre de sauts nécessaires est équivalent à évaluer le temps nécessaire pour transmettre une requête. Dans la suite des simulations, nous considérerons seulement le cas $\alpha(m) = \alpha_2(m)$. Lorsque $N = 20$, la probabilité d'être découvert est en fait peu élevée comparée à la probabilité d'avoir une déconnexion. En effet, on voit clairement qu'un facteur de branchement faible permet d'obtenir les meilleures performances. Les autres graphiques montrent que plus il y a de nœuds, plus un facteur de branchement élevé est nécessaire pour obtenir les meilleures performances. Cependant $m = 7$ n'est pas plus mauvais que $m = 10$ dans les cas présentés ici. On peut aussi choisir le meilleur compromis entre plusieurs cas. Ici $m = 5$ est le meilleur compromis. De plus, 5, 6 ou 7 sauts sont suffisants pour obtenir la meilleur atteignabilité. Considérons la figure 5(b), $0.6 \times 50 = 30$ serveurs sont atteints en 6 sauts. Si 100 machines sont connectées à chaque serveur alors on peut calculer les temps nécessaires pour propager une requête de supervision (voir 5.2.3). Dans le cas de notre architecture, ce temps est $6 \times t_s + 100 \times t_h$. Si l'on veut que celui-ci soit meilleur il faut qu'il soit inférieur à $3000 \times t_h$ qui est le temps nécessaire pour une solution centralisée. On doit donc avoir $t_s < 483 \times t_h$ alors que l'on peut considérer que le temps pour envoyer un message à un serveur ou à une machine est équivalent, c'est-à-dire $t_s \sim t_h$. Il est donc clair que même si notre solution introduit des nœuds intermédiaires, elle permet de réduire le délai nécessaire pour transmettre une requête de supervision.

6.1.4. *Nombre de nœuds*

En fixant le nombre de sauts et en utilisant $\alpha_2(m)$, on peut étudier l'atteignabilité selon le nombre total de nœuds sur la figure 6. On distingue une première partie où les courbes diminuent doucement correspondant à atteindre tous les nœuds exceptés ceux qui sont découverts. A partir d'un certain point, les courbes diminuent rapidement. Cela dépend du facteur de branchement qui limite le nombre de nœuds atteints à cause de la probabilité de disponibilité. Ainsi l'atteignabilité est affectée par deux facteurs :

β (première phase de la courbe) et $\alpha(m)$ (seconde phase de la courbe). Ici $m = 5$ est une fois de plus un bon compromis.

6.2. *Paramètres des modèles pair-à-pair*

Pour pouvoir comparer équitablement le modèle IRC avec les modèles pair-à-pair, les paramètres $\alpha(m)$ et β doivent être correctement fixés. Comme nos précédentes conclusions se sont faites en se basant sur une moyenne de 100 clients par serveur IRC, les valeur des paramètres ont été ajustées de la manière suivante :

– β est fixé à $\frac{\beta_{IRC}}{100} = \frac{0.01}{100} = 0.0001$
– $\alpha(m) = e^{\frac{2-log(N)}{100}}$

6.3. *Slapper*

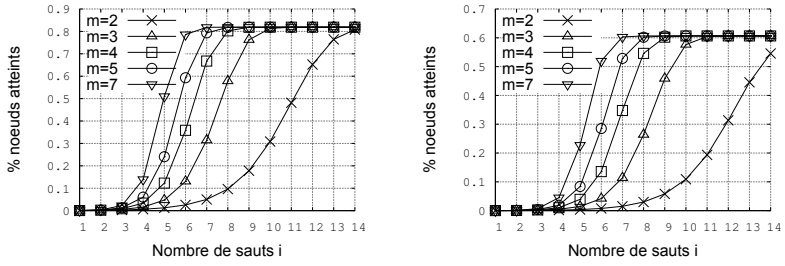
6.3.1. *Nombre de sauts*

Cette expérience cherche à mettre en évidence une relation entre l'atteignabilité et la distance. Bien entendu, on évalue l'atteignabilité selon plusieurs facteurs de branchement. On voit clairement apparaître une valeur maximale dans les figures 7(a) et 7(b). Ce seuil maximal est en fait directement dépendant de β similairement à ce que l'on a pu observer avec le modèle IRC. Cette valeur est donc une fois de plus la probabilité qu'aucun équipement ne soit découvert : $N \times (1 - \beta)^N$. Contrairement à IRC, on remarque ici qu'avoir un facteur de branchement m élevé est toujours bénéfique puisque plus celui-ci est élevé plus l'atteignabilité est meilleure. De plus, on remarque que quel que soit le nombre total d'équipements N, la distance nécessaire pour atteindre cette valeur maximale est toujours à peu près la même hormis pour un faible facteur de branchement. Par exemple avec $m = 5$, 8 sauts sont nécessaires. Ce genre d'information est très utile à un administrateur qui utiliserait un tel réseau pour configurer un ensemble d'équipements car il peut prévoir la distance et par extension le temps nécessaire à propager la requête à un ensemble d'équipements.

De plus, on voit que limiter la distance parcourue à 4/5 sauts est insuffisant en pratique. Enfin, bien que la distance nécessaire pour atteindre la valeur maximale soit fixe, le nombre total d'équipements a un effet sur la dynamique des courbes. En effet, quand N augmente, l'atteignabilité augmente plus au début pour des distances faibles alors qu'elle augmente moins rapidement pour les distances plus élevées.

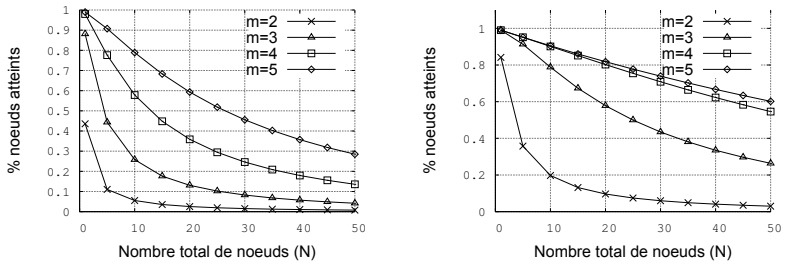
6.3.2. *Nombre de nœuds*

La dépendance entre l'atteignabilité et le nombre total de nœuds N est mis en évidence sur la figure 8. Pour cela nous considérons l'atteignabilité à 6, 8 et 10 sauts. Évidemment, l'atteignabilité diminue car malgré l'augmentation du nombre de nœuds, la distance est fixée ce qui ne permet pas de parcourir plus de nœuds. De plus, quand il



(a) N = 2000 (b) N = 5000

Figure 7. *Atteignabilité dans Slapper*



(a) i = 6 (b) i = 8

Figure 8. *Slapper, influence du nombre de sauts i*

y a de plus en plus de nœuds, les courbes avec les plus grands facteurs de branchement convergent vers les mêmes valeurs. Une fois de plus on retrouve ce seuil maximal qui est fixé par β, la probabilité qu'un nœud soit compromis. Cette observation confirme que pour un nombre de sauts suffisamment élevé, il est toujours possible d'obtenir la meilleure atteignabilité quel que soit le nombre total de nœuds. Si ce nombre n'est pas suffisamment élevé alors la différence est due au facteur de disponibilité $\alpha(m)$. D'un point de vue plus pratique, nous avons également noté qu'à partir de 10 sauts, le facteur de branchement n'a pas d'impact sur l'atteignabilité. Si on est à moins de 10 sauts alors on retrouve logiquement la relation mise en évidence dans l'expérience précédente : plus le facteur de branchement est élevé plus l'atteignabilité est élevée. Enfin, on remarque que pour un facteur de branchement $m = 2$, les performances sont très faibles et on ne peut pas appliquer les conclusions que l'on a déduit précédemment. Alors que cette valeur est la valeur par défaut du ver Slapper, elle semble être difficile à utiliser en pratique.

6.4. *Chord*

6.4.1. *Nombre de sauts*

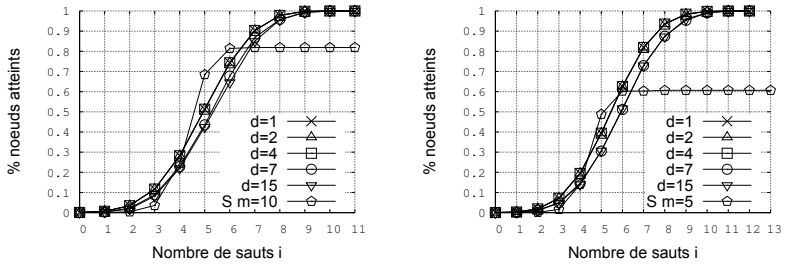
L'atteignabilité est tracée dans les figures 9(a) et 9(b) avec un nombre de nœuds différent. De manière générale on a $N_{MAX} = N \times d$ en considérant que l'on s'autorise à ce que ce ne soit pas une puissance de 2. Cela permet alors de comparer les résultats pour différentes distances d tout en gardant le même nombre de nœuds au total. On peut voir cela comme un anneau Chord qui aurait été tronqué à la fin. De ce fait, le nombre d'entrées dans la table de routage est le premier entier inférieur ou égal à $log(N_{MAX})$.

La première observation évidente est que toutes les courbes sont très proches entre elles. On pourrait d'ailleurs penser que les courbes avec les plus petites distances d sont les meilleures. Cependant il faut prendre en compte que d est parfois une puissance de 2, parfois pas. Dans l'équation [7], les termes $log(N_{MAX})$ et $log(d)$ devraient être des entiers. Dans notre expérience, ce n'est pas le cas et nous sommes alors obligés d'utiliser le premier entier inférieur ou égal à la valeur réelle. Ainsi on observe deux classes de courbes. La première, C_0, lorsque d est une puissance de 2 et on n'a donc pas de troncature. La seconde, C_1, où on a une troncature, ce qui réduit le nombre d'entrées de la table de routage et donc l'atteignabilité. Après avoir testé avec des valeurs de d de 1 à 16, on peut confirmer l'existence de ces 2 classes.

Pour un réseau de taille équivalente, la première classe est meilleure et l'atteignabilité est identique quel que soit d. Dans le second cas, l'atteignabilité est légèrement plus faible et on perçoit quelques différences entre chaque courbe. On observe que l'atteignabilité est au maximum diminuée de 10 points par rapport à la courbe de la classe C_0. De plus, l'atteignabilité maximale est proche de 1. Cela montre que, contrairement à IRC et Chord, β n'a pas un impact aussi grand car un attaquant qui voudrait connaître tout le réseau devrait dans le meilleur des cas (pour lui) pouvoir observer les connexions de $\frac{N}{log(N)}$ nœuds. Chord est donc la meilleure solution si le but est de propager une requête de management vers l'ensemble des équipements. Enfin, le cas correspondant à Slapper a été rajouté sur les figures 9(a) et 9(b) en considérant que le facteur de branchement était $log(N)$. On retrouve bien sûr une atteignabilité maximale plus limitée ce qui confirme notre conclusion précédente. Pour un nombre de sauts inférieur ou égal à 6, on peut considérer que Slapper est équivalent voire meilleur ce qui veut dire qu'utiliser Chord dans un tel cas de figure n'a pas d'intérêt.

6.4.2. *Distance d*

Dans Chord, le facteur de branchement est automatiquement fixé. On se propose alors d'étudier l'impact de la distance d entre 2 pairs consécutifs sur l'atteignabilité. En considérant $N_{MAX} = 2^{14}$ dans la figure 10(a), on remarque clairement que lorsque la distance augmente l'atteignabilité augmente. Cela est logique puisque lorsque d augmente, N décroît. La taille de la table de routage diminue aussi mais de manière moins significative puisque celle-ci a $log(N)$ entrées. De manière plus précise,



(a) Nombre de nœuds $N = 2000$ (b) Nombre de nœuds $N = 5000$

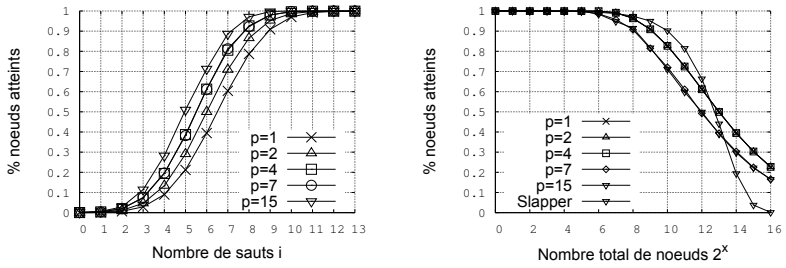
Figure 9. *Atteignabilité dans Chord (S = Slapper)*

lorsque d est multipliée par k, le nombre d'entrées de la table de routage diminue de $log(\frac{N_{MAX}}{d}) - log(\frac{N_{MAX}}{d \times k}) = log(k)$. Le nombre de nœuds dans le réseau est quant à lui divisé par k. Pour des valeurs assez élevées de N il est alors évident que le nombre de nœuds décroît plus vite que la taille de la table de routage. Finalement, même si le nombre de messages retransmis à chaque saut est plus petit, il y a beaucoup moins de nœuds à atteindre d'où de meilleurs résultats.

6.4.3. *Nombre de nœuds*

Comme pour les autres modèles, il est aussi possible d'étudier l'effet du nombre de nœuds sur l'atteignabilité. La Figure 10(b) considère une distance de 6 sauts et on remarque une fois de plus que les résultats sont très proches quel que soit d. En testant avec d'autres distances, on a conclu que lorsque l'on multiplie la distance d par $\frac{5}{4}$, la courbe est légèrement décalée à droite de 2^2 nœuds. Ainsi, si on avait une atteignabilité a pour N on la retrouve en $N + 2^2$. La figure montre une fois de plus que l'atteignabilité maximale est proche de 1. Logiquement comme la distance est fixée, lorsque le nombre de nœuds augmente, on ne peut plus tous les atteindre et les courbes diminuent.

Comme le facteur de branchement de Chord dépend directement de N, la courbe de Slapper correspond à l'atteignabilité maximale qui est d'ailleurs la même que pour un réseau IRC. La courbe représente donc la limitation de β soit $(1 - \beta)^N$. Il est clair qu'à partir de 2^{12} équipements à superviser, Chord est la meilleure solution. Pour un réseau de 2^{10} et 2^{12} Slapper ou IRC peuvent avoir de meilleurs résultats. En fait, on remarque sur les courbes 9(a) et 9(b) que 6 sauts correspondent à peu près à obtenir pour Slapper la meilleure atteignabilité ; correspond également à la limite après laquelle, les performances de Chord et Slapper se différencient grandement. Cependant il est clair que si on augmente la distance, Chord est bien meilleur même pour des valeurs de $N < 2^{12}$.



(a) Espace des ids $N_{MAX} = 2^{14}$ Nombre de sauts $i = 6$

Figure 10. *Atteignabilité dans Chord (S = Slapper)*

6.5. *Comparaisons générales*

6.5.1. *Impact des attaques*

Cette évaluation cherche à montrer la résistance des différents systèmes aux attaques. Comme il l'a déjà été dit, Slapper et Chord sont soumis de la même manière à la probabilité qu'un nœud soit attaqué β. Comme on peut considérer qu'un nœud compromis permet de connaître l'ensemble du réseau, soit un attaquant ne connaît pas du tout le réseau et dans ce cas cela signifie qu'aucun nœud n'est compromis avec donc une probabilité $(1-\beta)^N$, soit tout le réseau est connu avec une probabilité $1-(1-\beta)^N$. Le nombre moyen de nœuds connus par un attaquant est donc : $detect_{slapper} = N(1-(1-\beta)^N)$.

Dans le cas de Chord, le nombre moyen de nœuds attaqués est βN. Chacun de ces nœuds connaît $log(N)$ autres nœuds (formule [7]). On peut donc en déduire le nombre moyen de nœuds connus par un attaquant en considérant le pire cas où la table de routage de chaque nœud compromis est totalement différente. On obtient alors : $detect_{chord} = \beta N \times log(N)$.

On calcule le ratio $r(N) = \dfrac{detect_{slapper}}{detect_{chord}} = \dfrac{1-(1-\beta)^N}{\beta \times log(N)}$. Celui-ci ne dépend pas de la distance d. La figure 11 montre clairement l'effet bénéfique de Chord car le ratio augmente rapidement lorsque le nombre de nœuds dans le réseau augmente et atteint une valeur maximale $r(2^{15}) = 650$. Cela signifie qu'il y a 650 fois plus de nœuds connus avec IRC ou Slapper qu'avec Chord. On remarque ensuite que la courbe décroît et la limite en l'infini est $\lim_{N\to\infty} r(N) = 0$. Cependant pour un réseau de très grande taille, le ratio est de 20 avec 2^{512} nœuds. De manière générale Chord est plus résistant aux attaques que Slapper ou IRC car les nœuds de Chord n'ont qu'une vue partielle du réseau.

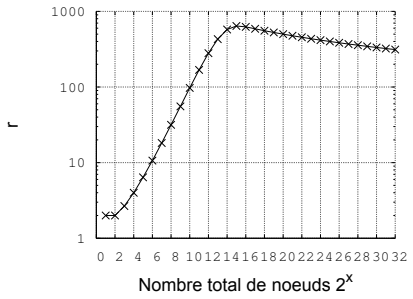


Figure 11. *Effet des attaques - $r(n)$*

6.5.2. *Facteur de branchement*

Il a été montré précédemment que contrairement à un réseau pair-à-pair, un facteur de branchement relativement élevé pouvait avoir des conséquences néfastes directes dans le cas d'un réseau IRC. Par exemple, on voit clairement que $m = 10$ n'est pas utilisable sur la figure 5(a). Les réseaux pair-à-pair sont donc résistants au problème de connectivité. Cependant, l'atteignabilité augmente généralement plus rapidement dans le cadre d'un réseau IRC comme sur les figures 5, 7(a), 7(b) et 9. Alors que pour $m = 7$, dans le cas de Slapper, 7 sauts sont nécessaires pour atteindre l'atteignabilité maximale, seulement 5 sont nécessaires pour atteindre la même atteignabilité maximale avec IRC. La figure 9(a) montre que 7 sauts sont aussi nécessaires dans le cas de Chord. En fait, un réseau IRC est beaucoup plus centralisé avec un nombre limité de serveurs ayant une grande connectivité (100 équipements par serveurs).

6.5.3. *Délais*

Considérons un temps de transmission t_{trans} entre deux nœuds quelconque (serveur ou équipements). Grâce aux formules [4] et [8] et en se basant sur les mêmes observations qu'au paragraphe précédent, on en déduit le délai nécessaire pour contacter tous les nœuds à 5 sauts dans le cas d'IRC $(5 + 100) \times t_{trans}$. Dans le cas d'un réseau pair-à-pair, on a $7 \times m \times t_{trans}$. Comme précédemment le cas équivalent est $m = 7$ (Slapper) soit le ratio suivant : $105/49$. Sur les figures 9(a) et (b), on remarque que les modèles P2P sont équivalents pour une distance inférieure à 6 sauts. Si l'application a des contraintes de temps fortes alors Slapper est une bonne solution. Si on veut de meilleures performances et notamment atteindre tous les nœuds, il est préférable d'utiliser Chord.

6.5.4. *Comparatifs*

Un bref récapitulatif des précédents résultats est présenté dans le tableau 1. L'anonymat du contrôleur est bien préservé dans les réseaux pair-à-pair grâce aux nœuds intermédiaires. Le faible nombre de ces derniers dans IRC ainsi que la nécessité d'avoir

	IRC	Slapper	Chord
Efficacité	plus petit nombre de sauts	plus courts délais	
Résistance	très contraints par une surcharge et par les attaques	très contraints par les attaques, peu de surcharge	résistance élevée, peu de surcharge, impact faible des attaques
Passage à l'échelle	$< 2^{12}$ nœuds		$\geq 2^{12}$ nœuds
Sécurité	le superviseur peut être détecté	difficile de retrouver le superviseur à cause des nœuds intermédiaires	
Intérêt	réseau large et fermé (une entreprise)	réseau large avec des participants connus et sûrs (honeypot distribué de recherche)	réseau ouvert et très large (honeypot public)

Tableau 1. *Comparaison des différentes solutions*

une autorité centrale prenant en charge le déploiement des serveurs rend ce type de réseau beaucoup plus vulnérable. L'avantage indéniable des réseaux IRC est que seul les serveurs ont besoin d'avoir un port ouvert. Les équipements à superviser n'ont pas alors à s'exposer à d'autres vulnérabilités en ouvrant un port. Ce type d'architecture est donc idéal pour une entreprise. Dans notre cas où l'on souhaite configurer des machines diverses déjà présentes sur un réseau pair-à-pair avec notamment des ports déjà ouverts, on peut penser que le botnet de supervision pourrait se baser sur les mêmes ports. Chord a quant à lui l'énorme avantage de passer beaucoup plus à l'échelle.

7. Conclusion et perspectives

Les applications malveillantes disposent de moyens de communication efficaces comme le montre le passage à l'échelle des botnets. Cet article propose une solution de supervision basée sur un botnet. Grâce à ces travaux, un administrateur peut savoir quelle est la performance du système selon la topologie par exemple. Selon les différentes contraintes qu'il peut y avoir, nous avons dressé un bilan comparatif des solutions. L'utilisation d'un réseau IRC permet de bonnes performances mais limite le passage à l'échelle et nécessite de mettre en place des serveurs. Les solutions pair-à-pair sont plus simples à mettre en œuvre tout en étant plus résistantes aux pannes ou aux déconnexions. Dans le cas précis de la configuration d'un honeypot public, il est clair que Chord est la meilleure solution envisageable avec notamment un passage à l'échelle plus important. Enfin, nous avons pour objectif de confronter cette évaluation à des tests réels en utilisant le réseau PlanetLab (Paterson *et al.*, 2006) comportant des points de présence variés sur internet. Cependant, pour pallier au nombre relativement faible de nœuds, il sera alors nécessaire de simuler plusieurs bots par nœud physique. De plus il sera nécessaire d'étudier la viabilité du système face à un attaquant qui essayerait de prendre le contrôle du botnet à son compte. Dans ce cas, il faudra évaluer

la résistance du botnet face à une attaque active où l'attaquant utilise ce qu'il a observé pour réussir à plus facilement s'introduire sur les différentes machines.

8. Bibliographie

Ahmed R., Boutaba R., « Distributed Pattern Matching : a Key to Flexible Efficient P2P Search », *IEEE Journal on Selected Areas in Communications*, vol. 25, n° 1, p. 73-83, January, 2007.

Arce I., Levy E., « An Analysis of the Slapper Worm », *IEEE Security and Privacy*, vol. 1, n° 1, p. 82-87, 2003.

Artigas M. S., López P. G., Gómez-Skarmeta A. F., « DECA : A Hierarchical Framework for DECentralized Aggregation in DHTs. », *DSOM*, p. 246-257, 2006.

Badonnel R., State R., Chrisment I., Festor O., « A Management Platform for Tracking Cyber Predators in Peer-to-Peer Networks », *icimp*, vol. 0, p. 11, 2007.

Badonnel R., State R., Festor O., « Probabilistic Management of Ad-Hoc Networks », *10th IEEE/IFIP Network Operations and Management Symposium - NOMS 2006*, IEEE Computer Society, 2006.

Barford P., Yegneswaran V., *An inside look at Botnets*, Springer, chapter 1, 2006.

Dabek F., Kaashoek M. F., Karger D., Morris R., Stoica I., « Wide-area cooperative storage with CFS », *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, October, 2001.

El-Ansary S., Alima L. O., Brand P., Haridi S., « Efficient Broadcast in Structured P2P Networks. », *IPTPS*, p. 304-314, 2003.

Goldszmidt G., Yemini Y., « Distributed Management by Delegation », *15th International Conference on Distributed Computing Systems*, IEEE Computer Society, 1995.

Grizzard J. B., Sharma V., Nunnery C., Kang B. B., Dagon D., « Peer-to-Peer Botnets : Overview and Case Study », *First workshop on Hot Topics in Understanding Botnets (HotBots 07)*, Cambridge, MA, April, 2007.

Lim K.-S., Stadler R., « A navigation pattern for scalable internet management », *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM'01), Seattle, USA, May 2001*, 2001.

Lim K.-S., Stadler R., « Real-time views of network traffic using decentralized management », *Integrated Network Management, 2005. 9th IFIP/IEEE International Symposium on*, 2005.

Maxima, « Maxima 5.14, a computer algebra system.
http ://maxima.sourceforge.net/index.shtml (accessed on 03/31/08) », 2007.

McLaughlin L., « Bot Software Spreads, Causes New Worries », *IEEE Distributed Systems Online*, 2004.

Ohno H., Shimizu A., « Improved network management using NMW (Network Management Worm) System », *Proceedings of INET'95 : Honolulu, Hawai'i, June 27-30, 1995.*, 1995.

Oikarinen J., Reed D., « rfc 1459 : Internet Relay Chat Protocol », 1993.

Paterson L., Roscoe T., « The Design Principles of PlanetLab », *Operating Systems Review*, vol. 40, n° 1, p. 11-16, January, 2006.

Provos N., « A virtual honeypot framework », *SSYM'04 : Proceedings of the 13th conference on USENIX Security Symposium*, USENIX Association, Berkeley, CA, USA, p. 1-1, 2004.

Ramachandran K., Sikdar B., « Modeling malware propagation in Gnutella type peer-to-peer networks », *International Parallel and Distributed Processing Symposium, 2006*, 2006.

Schiller C., Binkley J., *Botnets : The Killer Web Applications*, Syngress Publishing, 2007.

Schwartz B., Jackson A. W., Strayer W. T., Zhou W., Rockwell R. D., Partbridge C., « Smart packets : applying active networks to network management », *ACM Transactions on Computer Systems*, vol. 18, n° 1, p. 67-88, 2000.

SNMP, Research, « The Mid-Level Manager.
http ://www.snmp.com/products/mlm.html (accessed on 07/30/07) », 2007.

State R., Festor O., « Malware : a future framework for device, network and service management », *Journal in Computer Virology*, vol. 3, n° 1, p. 51-60, 2007.

Stoica I., Morris R., Karger D., Kaashoek F., Balakrishnan H., « Chord : A Scalable Peer-To-Peer Lookup Service for Internet Applications », *Proceedings of the 2001 ACM SIGCOMM Conference*, p. 149-160, 2001.

Szor P., *The Art of Computer Virus Research and Defense*, Addison-Wesley Professional, 2005.

Wang P., Sparks S., Zou C. C., « An Advanced Hybrid Peer-to-Peer Botnet », *First workshop on Hot Topics in Understanding Botnets (HotBots 07)*, Cambridge, MA, April, 2007.

Article reçu le 14 avril 2008
Accepté le 24 juillet 2008

Jérôme François *termine un doctorat en Informatique à l'université de Nancy. Le domaine de sa thèse est la modélisation des malware et plus particulièrement des botnets. Le but étant de mieux les comprendre pour les utiliser à des fins bienveillantes ou alors de mieux pouvoir s'en défendre.*

Radu State *est chercheur à l'INRIA Nancy - Grand Est dans l'équipe-projet MADYNES. Ses activités de recherche portent sur la sécurité de la voix sur IP et le fuzzing prôtocolaire. ll a publié plus de 50 articles en gestion de réseaux et sécurité. Il est membre de plusieurs comités de programme techniques et d'organisation : IEEE/IFIP Integrated Management, IEEE/IFIP NOMS, IEEE/IFIP DSOM et IPTCOMM.*

Olivier Festor *est directeur de recherche à l'INRIA Nancy - Grand Est où il dirige l'équipe-projet MADYNES. Ses activités de recherche portent sur la conception d'algorithmes et modèles pour de la gestion de la sécurité de réseaux et services de grande taille. Ses travaux incluent le monitoring, le fuzzing et l'analyse de vulnérabilités. Les domaines d'application sont les réseaux dynamiques, les services voix sur IP et IPv6. Membre du NMRG de l'IRTF et co-chair du groupe de travail 6.6 à l'IFIP, il a publié plus de 70 articles en gestion de réseaux et sécurité. Il est membre de plusieurs comités de programme techniques et d'organisation. Depuis 2006, il dirige le réseau d'excellence européen EMANICS portant sur les solutions de supervision pour l'internet du futur.*