Les botnets et la supervision à large échelle

Jérôme François, Radu State, Olivier Festor

Les botnets et la supervision à large échelle

Jérôme François, Radu State, and Olivier Festor

MADYNES - INRIA Nancy Grand Est, CNRS, Nancy-Université, 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 (formant le botnet) à travers le monde malgré les pare-feux, les détecteurs d'intrusions ou les autres équipements de sécurité. D'un point de vue technique, il serait très utile de les utiliser dans le cadre de la supervision des réseaux. Ce papier propose une nouvelle solution de supervision basée sur les botnets et évalue les performances associées grâce à un modèle analytique.

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 tel 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 (réseau 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 ce papier 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. La section 2 introduira les moyens de communications des applications malveillantes et comment il est possible de les adapter pour la supervision. Le modèle mathématique est présenté dans la section 3 avant que les résultats expérimentaux ne soient détaillés dans la section 4. La section 5 est un état de l'art. Enfin, nous conclurons et présenterons les pistes pour nos futures recherches.

2 Applications malveillantes adaptées pour le supervision

2.1 Moyens de communication des applications malveillantes

Il existe plusieurs types de logiciels malveillants dont les vers. Même si un vers 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.

2.2 Architecture de supervision

Dans une étude précédente [1], 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[2].

Un des moyens les plus connu est l'utilisation d'un réseau IRC dont le but initial est de permettre aux utilisateurs de discuter entre eux. En fait, le réseau est composé de plusieurs serveurs inter-connectés 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 équivalent 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.

Les précédentes observations laissent penser qu'un modèle de supervision des réseaux basé sur un botnet IRC est possible. On considère donc considérer que les clients (bots) sont les équiepements à superviser et que le superviseur est un client particulier (botmaster). Il est donc nécessaires de mettre en place des serveurs. Il est maintenant primordial d'évaluer les performances des botnets avant d'utiliser ce type d'intergiciel. 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. Il y a plusieurs questions que l'on peut se poser comme par exemple :

- quelle topologie dois-je utiliser pour mon réseau IRC ?
- combien de machines peut-on superviser ?
- combien de temps me faut-il pour propager une requête de supervision à 80% des équipements ?

Ceci vise à déterminer si notre solution de supervision est viable et passe à l'échelle ou non. Nous ne considérons pas la phase de déploiement du réseau IRC. L'arbre des serveurs est donc considéré construit dans la suite de notre analyse.

3 Modèle mathématique d'un botnet IRC

L'objectif du modèle analytique est de pouvoir prouver l'efficacité d'un botnet autrement que par l'expérience des botnets malveillants. Comme précédemment introduit, un botnet IRC est composé de plusieurs serveurs formant un arbre de diffusion. Pour l'instant nous ne considérerons que les serveurs que l'on nommera aussi noeuds. L'objectif est donc de calculer le nombre moyens de serveurs qui sont atteints à une distance données (sauts applicatifs). On commence donc par caluler le nombre moyens de voisins directs puis on généralise la formule.

De plus, 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. On considère qu'il y a N serveurs IRC. De manière à se rapprocher le plus possible de la réalité, nous intégrons un facteur de disponibilité : $\alpha(m)$. En effet, plus un serveur doit maintenir de connexions avec d'autres plus il a de chance d'être surchargé et donc d'être hors-service. $\alpha(m)$ est donc une fonction décroissante.

La probabilité p_k est la probabilité pour un noeud 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 distribtuion reflète que nous considérons qu'il n'y a apparement aucune topologie spécique 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 noeuds à une distance j d'une manière similaire à [3] :

$$G^j(x) = \begin{cases} G_0(x) \text{ pour } j = 1 \\ G^{j-1}(G_1(x)) \text{ pour } j \geq 2 \end{cases} \tag{1}$$

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

On peut alors calculer le nombre moyen de noeuds à une distance j du noeud d'origine qui est $z_j = (G^j)'(1)$. Une fois de plus, en utilisant une méthode similaire à [3], 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 :

$$G_0'(x) = \frac{\alpha(m)}{m} \sum_{k=1}^{m} k x^{k-1} \qquad G_0''(x) = \frac{\alpha(m)}{m} \sum_{k=1}^{m} k(k-1) x^{k-2}$$

3.1 Atteignabilité

L'atteignabilité est la proportion moyenne de serveurs atteints à une distance k donnée 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 interessé pour compromettre le botnet de manière à récupérer des informations sensibles sur le réseau et les équipements de l'entreprise. C'est pour cette raison qu'il est nécessaire de définir un nouveau paramètre β qui est la probabilité pour un serveur d'être compromis, c'est à dire que l'attaquant peut surveiller tous les échanges réseaux et donc découvrir les autres serveurs c'est à dire connaître leurs addresses IP. On peut donc considérer que le réseau reste opérationnel si aucun des noeuds n'est compromis. On multiplie donc la proportion de noeuds atteints par la probabilité d'avoir le réseau non découvert qui est $(1-\beta)^N$ c'est à dire qu'aucun des N noeuds n'est compromis.

L'atteignabilité peut alors être complètement définie :

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

L'atteignabilité des bots 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 : $att_{bots}(k) = \dfrac{bots(k)}{B} = att(k)$

3.2 Atteignabilité moyenne

De manière à obtenir un indicateur globale 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.

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

3.3 Charge du système

La charge d'un serveur est une autre métrique importante qui permet de savoir si déployer un tel système est viable. Il faut considérer la charge $charge_s$ ou les ressources nécessaires pour maintenir une connexion avec un serveur ainsi que

la charge $charge_c$ pour maintenir une connexion avec un équipement final. Une solution classique centralisée est composée d'une seule plateforme de supervision. Dans ce cas la charge totale est seulement composée de la charge pour maintenir les connexions avec les équipements puisque qu'il n'y a aucun autre serveur : $charge\_serveur_{centralise} = C \times charge_c$.

Dans le cas des botnets, la charge d'un serveur est la charge pour maintenir $C_{serveur}$ connexions avec les équipements finaux ainsi qu'avec les autres serveurs où $C_{serveur}$ est le nombre moyen d'équipements connectés à un même serveur IRC. On obtient donc la charge d'un serveur grâce à la formule suivante : $charge\_serveur_{botnet} = C_{serveur} \times charge_c + m \times charge_s$.

3.4 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 : $temps_{centralise} = C \times t_h$.

Dans le cas d'un botnet, les serveurs peuvent transmettre les messages aux serveurs avant de les transmettre aux machines. Le temps total est donc composé du temps pour atteindre le dernier serveur plus le temps pour ce serveur de transmettre les requêtes à l'ensemble des machines connectées : $temps_{botnet} = k \times t_s + C_{serveur} \times t_h$.

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

4 Résultats expérimentaux

L'objectif de cette section est de déterminer les performances d'une architecture de management basée sur les botnets selon différentes configuration. 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 utilisés des outils mathématiques Maxima [4].

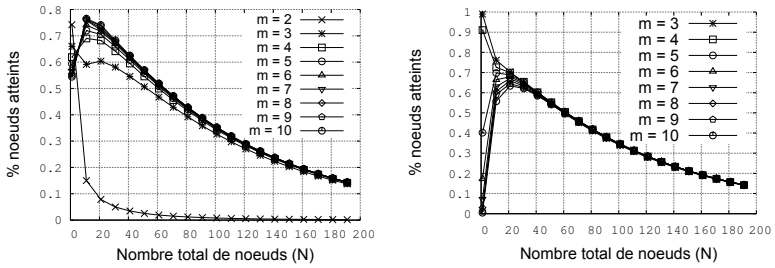
4.1 Paramètres

$\alpha(m)$ 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 noeuds 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 $i = 3$. La probabilité pour un noeud d'être découvert est $\beta = 0.01$. Ces paramètres ont été fixé arbitrairement mais représente quand même le système dans un environnement hostile car par exemple pour seulement 20 serveurs, *beta* induit une probabilité d'être totalement découvert de 20%. Ces paramètres doivent être en réalité adapté aux contraintes spécifiques à l'environnement où est déployé le système.

4.2 Atteignabilité moyenne



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

Fig. 1. Atteignabilité moyenne selon différents facteurs de branchement

Dans un premier temps, l'atteignabilité moyenne sera évaluée selon différent facteurs de branchement et le nombre total de serveurs/noeuds dans le réseau. Sur la figure 1(a), m varie de 1 à 10 et $\alpha_1(m)$ est utilisé. Au début, comme il y a de plus en plus de noeuds dans le réseau, plus de noeuds 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 *beta* limite le nombre de noeuds atteints. On en déduit qu'en terme d'atteignabilité, déployer un réseau de plus de 10 noeuds 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 noeud 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 1(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 dépends de $\alpha(m)$ alors que la seconde partie, ici après 20 noeuds, dépend de β.

Considérons maintenant le nombre de noeuds atteints et non plus la proportion correspondante. On remarque sur la figure 2(a) que toutes les courbes sont similaires et qu'elles atteignent une valeur maximale de 35 noeuds pour 100 noeuds 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 financière imposées.

Par exemple, avec 100 bots connectés à un serveur, il est possible dans le meilleur des cas de superviser 3500 machines. On peut alors calculer les charges des différents systèmes grâce aux formules de la section 3.3 :

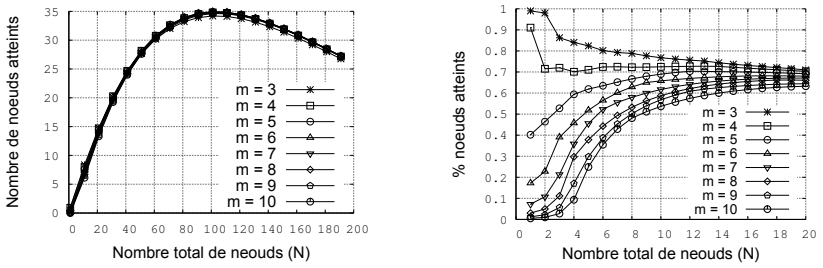
$$charge\_serveur_{centralise} = 3500 \times charge_c$$

$$charge\_serveur_{botnet} = 100 \times charge_c + m \times charge_s$$

Notre but est de diminuer la charge d'un serveur et donc d'avoir :

$$charge_s < \frac{3400}{m} charge_c$$

On considère que la charge pour maintenir une connexion avec un autre serveur est supérieure à celle nécessaire pour maintenir une connexion avec une machine cliente d'où $charge_s > charge_c$. Même si l'on considère un facteur de branchement de 10, $charge_s$ peut être égale à $340 \times charge_c$ pour obtenir des performances équivalentes ou meilleures. 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 2(b). L'atteignabilité dépend de deux forces antagonistes : le facteur de branchement et la probabilité d'avoir une machine en panne c'est à dire $1-\alpha(m)$.

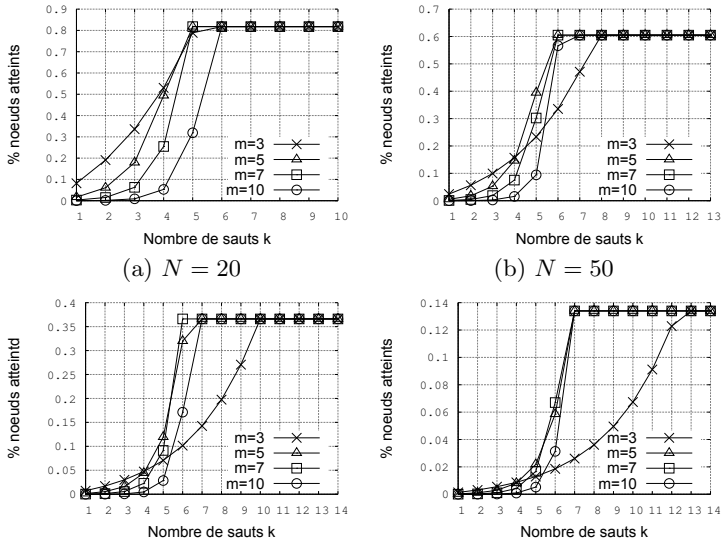


(a) Nombre absolu de noeuds atteints (b) Atteignabilité moyenne avec $1 \leq N \leq 20$

Fig. 2. Noeuds atteints avec $\alpha(m) = \alpha_2(m)$

4.3 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 3(a), 3(b), 3(c) et 3(d). On 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



(a) $N = 20$ (b) $N = 50$

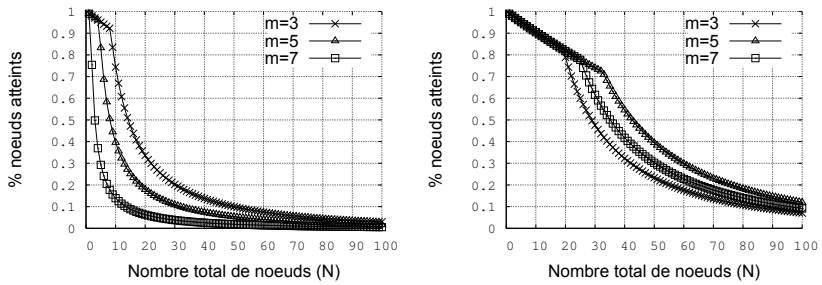
(c) $N = 100$ (d) $N = 200$

Fig. 3. Dépendance entre atteignabilité et facteur de branchement

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 noeuds, 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 3(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 nécessaire pour propager une requête de supervision (voir 3.4). 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 noeuds intermédiaires, elle permet de réduire le délai nécessaire à transmettre une requête de supervision.

4.4 Nombre de noeuds

En fixant le nombre de sauts et en utilisant $\alpha_2(m)$, on peut étudier l'atteignabilité selon le nombre total de noeuds sur la figure 4. On distingue une première



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

Fig. 4. Atteignabilité selon N et le nombre de sauts

partie où les courbes diminuent doucement correspondant à atteindre tous les noeuds en ayant pour seule limitation la probabilité d'être découvert. A partir d'un certain point, les courbes diminuent rapidement. Cela dépend du facteur de branchement qui limite le nombre de noeuds 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.

5 Etat de l'art

Dans une solution classique de supervisions le superviseur communique directement avec les équipements concernés. D'autres solutions sont ensuite apparues comme la supervision par délégation [5] ou encore l'utilisation de superviseurs en cascade [6]. Un superviseur intermédiaire adapte les requêtes aux équipements dont il est responsable. Les réseaux actifs sont une autre possibilité [7] mais exigent des équipements dédiés. Dans [8], une solution spécifique à la distribution des mises à jour Windows s'appuie sur un regroupement efficace des mise à jours. Pour les réseaux ad-hoc, une solution dédiée est présentée dans [9] où les machines avec une forte connectivité sont déterminées de manière à élire des superviseurs. Dans [10], un vers se promène sur le réseau, récolte des informations et retourne vers un superviseur. Contrairement à ces solutions, nous proposons d'utiliser un botnet composé quelques serveurs qui doivent seulement transmettre les requêtes. De plus, les problèmes relatifs aux parefeux sont résolus car ce sont les équipements supervisés qui doivent se connecter au réseau IRC. De nombreux détails à propos des botnets se trouvent dans [11]. Plusieurs familles de botnets existent [12]. Elle utilise classiquement un réseau IRC qui limite la latence et préserve l'anonymat du contrôleur [13]. Nous sommes les premiers à proposer un modèle analytique pour vérifier expérimentalement les performances des botnets.

6 Conclusion et perspectives

Les applications malveillantes disposent de moyens de communication efficaces comme le montre le passage à l'échelle des botnets. Ce papier décrit un modèle analytique précis d'un botnet IRC dans le contexte de la supervision des réseaux. Grâce à ces travaux, un administrateur peut savoir quelle est la performance du système selon la topologie par exemple. De plus, le réel objectif est de déterminer le nombre de serveurs ainsi que la topologie (facteur de branchement) en prenant en compte les différentes contraintes : délai, nombre d'équipements à superviser... Nous allons par la suite tester une implémentation pour configurer un honeypot distribué dont le but est de repérer les criminels sur les réseaux pair-à-pair d'échange de fichiers [14]. **Remerciements** Ce papier est en partie soutenu par EC IST-EMANICS Network of Excellence (#26854).

Références

1. State, R., Festor, O. : Malware : a future framework for device, network and service management. Journal in Computer Virology **3**(1) (2007) 51–60
2. McLaughlin, L. : Bot software spreads, causes new worries. IEEE Distributed Systems Online **5**(6) (2004)
3. Ramachandran, K., Sikdar, B. : Modeling malware propagation in gnutella type peer-to-peer networks. International Parallel and Distributed Processing Symposium, 2006 (2006)
4. Maxima : A computer algebra system.
http ://maxima.sourceforge.net/ (accessed on 12/05/07)
5. Goldszmidt, G., Yemini, Y. : Distributed management by delegation. In : 15th International Conference on Distributed Computing Systems, IEEE Computer Society (1995)
6. SNMP, Research : The mid-level manager.
http ://www.snmp.com/products/mlm.html (accessed on 07/30/07)
7. Brunner, M., Stadler, R. : Management in telecom environments that are based on active networks. Journal of High Speed Networks (2001)
8. Gkantsidis, C., Karagiannis, T., VojnoviC, M. : Planet scale software updates. SIGCOMM Comput. Commun. Rev. **36**(4) (2006) 423–434
9. Badonnel, R., State, R., Festor, O. : Probabilistic management of ad-hoc networks. In : 10th IEEE/IFIP Network Operations and Management Symposium - NOMS 2006, IEEE Computer Society (2006)
10. Ohno, H., Shimizu, A. : Improved network management using nmw (network management worm) system. In : Proceedings of INET'95 : Honolulu, Hawai'i, June 27-30, 1995. (1995)
11. Nazario, J. : Defense and Detection Strategies against Internet Worms. Artech House, Inc., Norwood, MA, USA (2003)
12. Barford, P., Yegneswaran, V. : 1. In : An inside look at Botnets. Springer (2006)
13. Cooke, E., Jahanian, F., Mcpherson, D. : The zombie roundup : Understanding, detecting, and disrupting botnets. (June 2005) 39–44
14. Badonnel, R., State, R., Chrisment, I., Festor, O. : A management platform for tracking cyber predators in peer-to-peer networks. icimp **0** (2007) 11