Conception d'une plateforme expérimentale pour l'étude comportementale de maliciels de type bots

Ion Alberdi et Philippe Owezarski et Vincent Nicomette

LAAS-CNRS
7 avenue du Colonel Roche - 31077 Toulouse Cedex 4
{*ialberdi,owe,nicomett*}*@laas.fr*

Protéger les réseaux contre les attaques de déni de service distribuées (DdSD) est une des problématiques stratégiques principales de l'Internet. Les solutions courantes à base de systèmes de détection d'intrusions présentent l'inconvénient d'être postactives (ou réactives) et donc de ne fonctionner qu'a posteriori. Les systèmes de défense de l'Internet devraient plutôt être proactifs, i.e. capables d'empêcher les attaques de se produire. Cet article, qui s'inscrit dans cette optique de recherche à long terme, propose, dans ce but, de commencer par observer et analyser les attaques et les pirates. En particulier, l'objectif est de détecter et d'infiltrer ce qui apparaît comme la source principale des attaques, à savoir les `botnet`s. Notre étude se concentre sur l'étude comportementale de certains maliciels, à savoir ceux qui pilotent les `bot`s. Ce papier présente la conception et la mise en place d'une plateforme expérimentale qui utilise comme technique de métrologie/supervision la technologie des « pots de miel » pour télécharger les maliciels, mais surtout, et c'est la principale contribution de ce papier, une infrastructure permettant l'observation et l'analyse du comportement des maliciels collectés, que nous considérons comme étant un pot de miel post infection. Ce genre de plateforme est confronté à des problèmes de légalité par rapport à des risques de pollution des systèmes d'information d'autrui. Notre approche a donc pour contrainte de réduire au maximum le risque d'envoyer du trafic malicieux vers l'Internet. Ce système a été mis en place, puis testé durant quatre mois. Ce papier présente les premiers résultats obtenus par l'observation des maliciels collectés, leurs premières analyses et les premières conclusions sur le comportement des `botnet`s, démontrant ainsi l'intérêt et la pertinence de la plateforme.

Mots-clés: pot de miel, `botnet`, maliciel.

1 Introduction

Alors que l'Internet est en train de devenir un réseau multi-services pour différents types d'applications aux besoins divers, les attaques de Déni de Service Distribuées (DdSD) représentent une menace d'importance. En effet, garantir la Qualité de Service est devenu aujourd'hui un des leitmotiv de l'Internet qui en plus de transporter des fichiers, transporte également de la voix et de la vidéo (pour ne citer que ces types de média) ayant des contraintes temporelles fortes. Ainsi, il suffit qu'une attaque ralentisse un peu le réseau, pour que la qualité du service ne soit plus suffisante, et le service par conséquent non rendu (et donc non facturable).

Ces types d'attaques sont extrèmement fréquentes. L'exemple le plus célèbre est certainement l'attaque lancée le 17 Octobre 2002 contre les 13 serveurs DNS racines de l'Internet. Cette attaque avait réussi à rendre 7 de ces serveurs indisponibles, et une partie non négligeable de l'Internet s'est vue privée de ce service durant une période significative, engendrant des délais péniblement longs pour les utilisateurs. De façon générale, [MMS01] a dénombré plus de 12000 attaques ciblant 5000 hôtes différents durant une période de 3 semaines en 2001. [Sym06] affirme que de janvier à juin 2006 il y a eu une moyenne de 66.000 `bots`[†] qui ont lancé 6110 attaques par jour.

[†]Machine compromise pilotée à distance.

Protéger les réseaux contre ce type d'attaques est une des problématiques stratégiques principales à résoudre. Les solutions courantes de protection se basent généralement sur l'utilisation des systèmes de détection d'intrusion. L'inconvénient de ces approches est d'être postactives (ou réactives) et donc de ne fonctionner que lorsque les attaques ont lieu [ZP05]. Les systèmes de défense de l'Internet doivent donc évoluer de façon à être proactifs, i.e. être capables d'empêcher les attaques de se produire. Cet article s'inscrit dans cette optique de recherche à long terme qui n'en est qu'à ses balbutiments. Plus précisément, nous proposons d'utiliser des techniques de métrologie/supervision pour observer et analyser les attaques et les pirates. En particulier, l'approche proposée dans ce papier a pour but de détecter et d'infiltrer ce qui apparaît comme la source principale des attaques, à savoir les `botnets`[‡] [FHW05]. Les techniques de métrologie utilisables peuvent être des outils classiques de métrologie passive des réseaux, mais également dans le cadre des attaques, des outils plus spécialisés appelés pots de miel.

Pour détecter ces `botnet`s, on peut distinguer trois sources d'informations :

1. L'observation du canal `Command & Conquer`, (C&C) qui est le canal de contrôle des machines corrompues par le pirate.
2. L'observation du trafic malicieux envoyé par les `bot`s (flood, spam, informations confidentielles récupérées sur les machines, ...).
3. Le maliciel qui corrompt un ordinateur et le transforme en bot.

Notre travail se concentre sur l'étude comportementale de ce type de maliciels [Hol05], ces derniers étant les logiciels qui pilotent les `bot`s. Ce papier présente la conception et la mise en place d'une plateforme expérimentale qui utilise la technologie des « pots de miel » pour télécharger les maliciels , mais surtout, et c'est la principale contribution de ce papier, une infrastructure permettant l'observation et l'analyse du comportement des maliciels collectés (partie 3 pour la description du principe de la méthode et partie 4 pour son implémentation). Ce genre de plateforme est confronté a des problèmes de légalité évidents par rapport aux risques de pollution des systèmes d'information d'autrui[§]. Notre approche a l'originalité de fonctionner dans un environnement restrictif, la principale contrainte étant d'interagir avec l'extérieur en réduisant au maximum le risque d'envoyer du trafic malicieux vers l'Internet. Ce système a été mis en place, puis testé durant quatre mois. Ce papier présente donc les premiers résultats obtenus par l'observation des maliciels collectés, leurs premières analyses et les premières conclusions sur le comportement des `botnet`s, démontrant ainsi l'intérêt et la pertinence de la plateforme (partie 5). Enfin, la partie 6 conclut le papier en décrivant certains aspects qui doivent être améliorés dans de futurs travaux. Elle illustre également par quelques exemples comment ces résultats peuvent être exploités pour la conception d'un système de protection proactif global dans l'Internet.

2 État de l'art sur l'étude des botnets

Comme décrit dans l'introduction, trois sources d'informations peuvent être utilisées pour détecter des `botnet`s.

L'exploitation de la première source d'information, à savoir la détection de canaux de type C&C a donné lieu à plusieurs travaux [AKS+06, Kri04, Rac04]. Ces études analysent ce trafic, et identifient les `botnet`s en observant des commandes IRC non standard ou en mesurant la réactivité des clients IRC pour différencier les `bot`s des humains. Ces travaux présentent l'inconvénient de se réduire à un seul type de canal C&C, à savoir l'IRC. Ces études ont de plus supposé que le trafic IRC utilisé par les `botnet`s se réduisait au trafic IRC sur le port standard, ce que réfute par exemple [CJM05].

À notre connaissance l'étude du trafic malicieux ou des vecteurs d'attaques (2^{nd} source d'information) n'a pas eu d'application dans la détection de `botnet`s. Cette piste reste donc à explorer.

[‡]Pour justifier cette affirmation, l'étude [Mas06] affirme que la location de `botnet`s (utilisé en particulier pour le lancement de DdSD ou de spam) est une activité très rentable, et selon leurs estimations, les profits engendrés par ces pirates seraient équivalents à ceux de l'industrie des logiciels antivirus.

[§]Le propriétaire d'une machine est responsable du trafic qu'elle génère. Ainsi, le propriétaire d'une machine corrompue est responsable des trafics d'attaques que le pirate qui en a pris le contrôle lui fait générer.

L'exploitation de la troisième source d'informations (les maliciels qui corrompent les ordinateurs et les transforment en bots) a donné lieu à plusieurs travaux. Ces travaux utilisent la technologie des « pots de miel » que l'on définit comme :

Définition 1 *Un « pot de miel » est une ressource d'un système d'information dont la valeur réside dans son usage non autorisé ou illicite.*[¶]

Les pots de miels traités dans ce papier sont utilisés pour simuler une activité malicieuse : tantôt une réussite d'attaque, tantôt les agissements malicieux post infection.

Nous pouvons différencier deux types de « pot de miel » :

1. Les pots de miels dits de haute interaction : ils simulent entièrement une machine vulnérable, en général des systèmes d'exploitation complets sont utilisés pour les implémenter.
2. Les pots de miels dits de basse interaction : ils simulent une partie du comportement de la machine vulnérable, idéalement le strict minimum nécessaire pour faire croire à l'assaillant que son attaque a réussi.

[HP05] décrit les avantages et inconvénients des deux types de pots de miel. Les hautes interactions simulent entièrement un service vulnérable et ont donc plus de chance de tromper un assaillant. Cependant, l'utilisation de ce type de « pot de miel » présente des risques et demande donc un investissement humain assez important. Par conséquent, leur utilisation est péniblement automatisable. Inversement les pots de miel basses interactions sont plus facilement administrables, mais présentent l'inconvénient de ne pas toujours correctement interpréter certaines attaques. Les auteurs de [HP05] préconisent donc l'utilisation conjointe des deux technologies : les pots de miel à haute interaction servent à calibrer les pots de miel à basse interaction qui permettent d'exécuter des tâches automatisables.

Afin d'étudier les botnets au moyen de l'analyse de maliciels, nous devons faire face à deux problématiques :

1. Comment se procurer un maliciel.
2. Comment analyser le maliciel, extraire les informations sur le `botnet` correspondant[‖] et simuler son exécution.

[CJM05] propose d'installer un « pot de miel » haute interaction dans un réseau et d'essayer de protéger ce réseau d'une éventuelle propagation en limitant la bande passante en sortie, et en filtrant les tentatives de connexions vers le réseau local. Ce système simule donc entièrement le comportement d'un bot. Cependant cette approche présente à nos yeux deux inconvénients majeurs :

1. Une difficulté d'administration liée à l'utilisation d'un pot de miel haute interaction pour la simulation d'une vulnérabilité.
2. Un manque de maîtrise de l'exécution du maliciel. Avec ce système nous ne pouvons pas :
   – Maîtriser la date à laquelle on simule l'exécution de ce maliciel.
   – Offrir des garanties sur la protection contre les dégâts potentiels que pourrait occasionner ce type de logiciels. Ce système nous assure que les octets émis n'atteindront pas le réseau local. Cependant en ce qui concerne le trafic à destination de l'Internet, même si la limitation de débit en sortie nous permet de réduire - mais pas d'empêcher - le dégâts des attaques de déni de service potentielles, il ne résoud pas les problèmes occasionnés par les tentatives de propagation du maliciel.

La méthodologie présentée dans [FHW05] vise à se faire passer pour un `bot` dans le but d'infiltrer un `botnet`. Pour ce faire, la première étape consiste à télécharger le pilote du `botnet`, à savoir le maliciel, l'analyser et en extraire les paramètres sur le canal C&C utilisé. Une fois ces paramètres obtenus la stratégie consiste à émuler le comportement d'un bot pour observer ce canal. Cette approche propose un logiciel de collecte de maliciels, plus automatisable que l'approche présentée dans [CJM05]. L'émulation du bot est moins précise que dans [CJM05] mais offre beaucoup plus de garantie. En effet, l'émulation de la communication sur le canal C&C plutôt que l'émulation de l'interprétation des ordres (comme dans [CJM05]), permet de ne pas envoyer de trafic malicieux vers l'Internet.

[¶] `http://www.honeynet.org`.

[‖] S'il fait partie d'un botnet.

3 Conception du système d'analyse des maliciels

Notre approche a pour but d'offrir un environnement d'émulation du comportement d'une attaque après infection, tout en limitant au maximum les paquets malicieux envoyés vers l'Internet. La contrainte que nous nous fixons de ne pas envoyer de paquets malicieux vers l'Internet, peut être discutable. Pourquoi ne pourrions nous pas envoyer certains paquets malicieux si, grâce à ces paquets, nous réussissons à considérablement réduire l'activité malicieuse de l'Internet ? La loi française a légiféré sur ce sujet et considère que ce trafic est illégal. Nous avons donc fait notre analyse en respectant ce cadre juridique.

Cette approche se base sur la résolution de deux problématiques principales :

– Comment se procurer des maliciels ?
– Comment créer l'environnement d'émulation ?

3.1 Comment se procurer des maliciels ?

Pour se procurer des maliciels, une des solutions consiste à se faire volontairement infecter, en utilisant des « pots de miels ». Nous allons dans cet article différencier deux types de maliciels, qui exploitent des vulnérabilités :

De type client exploitant des logiciels clients, nécessitant ainsi une interaction humaine de l'utilisateur (comme le clic sur une pièce jointe d'un logiciel de messagerie, la visite d'un site Web particulier, etc.).

De type serveur exploitant des bogues de logiciels serveurs et qui ne nécessitent donc aucune interaction humaine pour pouvoir être utilisés.

Un logiciel de collecte de maliciels obtenus par exploitation de bogues logiciels serveurs est disponible sous licence GNU : `nepenthes`. Ce logiciel est une évolution du travail commencé par [FHW05] avec le logiciel `mwcollect` et est décrit dans [BKH+06]. Nous avons choisi dans un premier temps d'étudier les maliciels collectés par ce logiciel.

3.2 Comment créer l'environnement d'émulation ?

L'objectif est maintenant d'exécuter les maliciels collectés afin d'en comprendre le fonctionnement. Pour cela, puisque nous ne pouvons pas légalement les exécuter sans restriction, nous avons créé un environnement de simulation. La qualité d'émulation des agissements de ce type de maliciel dépend du niveau d'interaction du système. En développant un client IRC particulier pour observer un `botnet` comme l'ont fait [FHW05], le niveau d'interaction obtenu est inférieur à celui que l'on peut espérer en exécutant ce maliciel. Un niveau d'interaction égal à celui des `bots` (c'est-à-dire une exécution sans contrôle) pose des problèmes quant à l'intégrité des systèmes d'information et au respect de la loi. Nous avons donc choisi de limiter ces effets de bord, au détriment du niveau d'interaction.

Les modifications que peut engendrer un tel maliciel sur un système d'exploitation peuvent être conséquentes et influencer le comportement des processus s'exécutant à posteriori, en particulier l'exécution d'autres maliciels. Nous pensons donc que le système dans lequel ce dernier va être exécuté doit être « propre », et avons choisi de n'exécuter qu'un seul maliciel à la fois, puis de réinitialiser le système.

La deuxième contrainte implique la maîtrise des communications sortantes. Le trafic du canal C&C n'est pas en lui même un danger pour le système d'information : il ne s'attaque aucunement au système sur lequel il s'exécute, il utilise simplement des ressources réseaux. Nous essayons donc de fournir un niveau d'interaction aussi haut que possible sur l'émulation de ces communications : laisser notre `bot` communiquer avec ce canal lorsqu'il est fonctionnel, et de le simuler sinon[**]. Le trafic malicieux potentiellement émis par le `bot`[††] étant quant à lui illégal, nous nous contentons de simuler ce type d'interaction.

Il reste à savoir comment identifier les flux et simuler certaines interactions. Nous utilisons pour cela une approche itérative en redirigeant les flux vers des machines que nous maîtrisons. Lorsque nous identifions un nouveau flux, nous le redirigeons dans un premier temps vers une machine de notre réseau local qui simule le service demandé par ce flux. On observe et on analyse ensuite ce flux pour décider de l'autoriser ou non lors de la prochaine exécution du maliciel.

[**] Par exemple dans le cas IRC : url périmée, serveur indisponible...

[††] DoS, tentatives de propagation, spam.

4 Implémentation

4.1 Implémentation de la plateforme de téléchargement

Nous avons utilisé une version modifiée de `nepenthes` pour collecter les maliciels. En effet les communications initiées depuis notre pot de miel vers Internet étaient problématiques. Or pour que ce pot de miel fonctionne correctement, toutes les communications qu'il initie doivent être autorisées[‡‡]. De plus les attaques ayant pour but de fournir un `shell` à l'attaquant imposent de devoir autoriser tous les flux en entrée de la machine[§§]. Ainsi la machine se doit d'être dépourvue de tout filtrage pour être totalement fonctionnelle. Nous n'avons cependant aucune garantie sur le fait que le logiciel `nepenthes` ne dispose pas lui aussi des bogues de type serveur exploitables. Nous avons donc finalement opté pour l'architecture de téléchargement décrite sur la figure 1. Nous avons coupé le logiciel `nepenthes` en deux :

1. Une partie qui interprète les attaques et en déduit les paramètres de demande de téléchargement.
2. Une partie qui, en fonction de ces paramètres, télécharge les virus.

Cette décomposition permet de fournir une politique de filtrage plus fine : n'autoriser aucun flux émis par le simulateur de vulnérabilité, et n'autoriser aucun flux en entrée de la partie responsable du téléchargement[¶¶].

L'échange d'informations relatives aux paramètres de téléchargement se fait par l'intermédiaire d'un journal produit par le simulateur de vulnérabilité et auquel le logiciel de téléchargement accède par le biais de `ssh/sftp` (`http://www.openssh.com/`), ou plus exactement, pour automatiser cette tâche, par `sshfs` (`http://fuse.sourceforge.net/sshfs.html`).

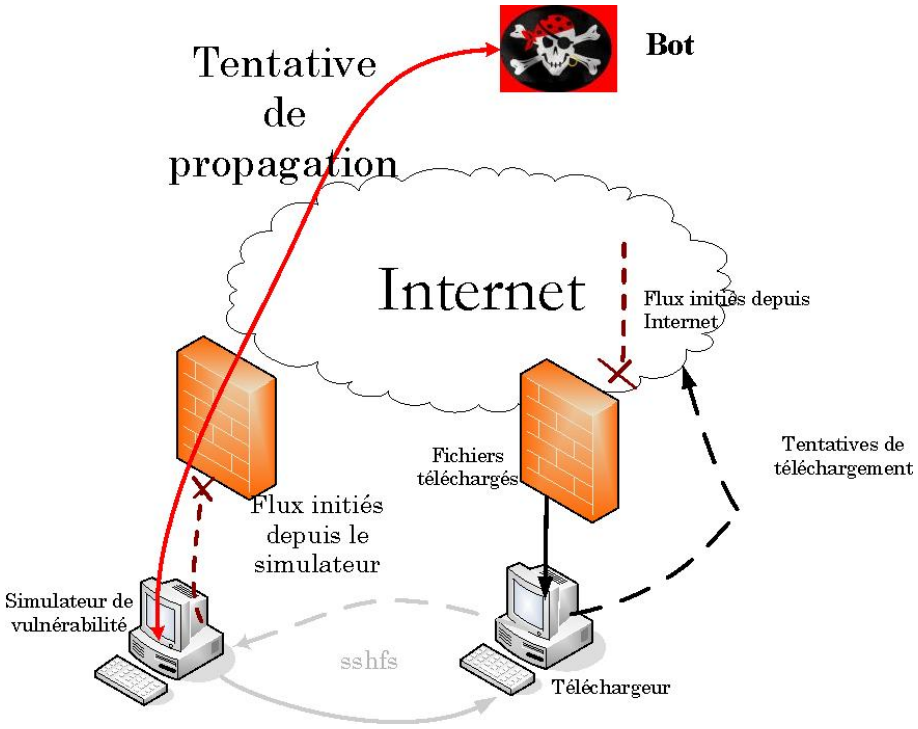

FIG. 1: Système de téléchargement

[‡‡] Car nous ne pouvons, par exemple, pas prévoir quelles seront les demandes de téléchargement.
[§§] Car nous ne pouvons pas anticiper sur quel port ce `shell` devra être associé (« bind »).
[¶¶] Toutes ces considérations sont à voir du point de vue d'un pare feu à état.

4.2 Implémentation de l'environnement d'exécution

La grande majorité des systèmes installés sur les ordinateurs interconnectés à l'Internet étant `Windows XP`, nous avons choisi de baser notre environnement d'émulation sur ce système. Pour obtenir un système propre à chaque exécution, le logiciel de simulation d'architecture x86 `VmWare` a été utilisé. Ce logiciel propose de pouvoir sauvegarder l'état courant de la machine virtuelle et d'y revenir quand bon nous semble. Pour avoir un système propre à chaque exécution, il nous suffit dès lors de mémoriser un état propre, et d'y revenir lors d'une nouvelle exécution.

Nous avons choisi le système `GNU/Linux` comme système hôte essentiellement pour son pare feu `netfilter` et en particulier l'API de programmation offerte par la librairie `libipq` qui permet de déléguer certains traitements à l'espace utilisateur. Nous l'avons aussi choisi car le logiciel de virtualisation le supporte.

Le système global de simulation est schématisé sur la figure 2.

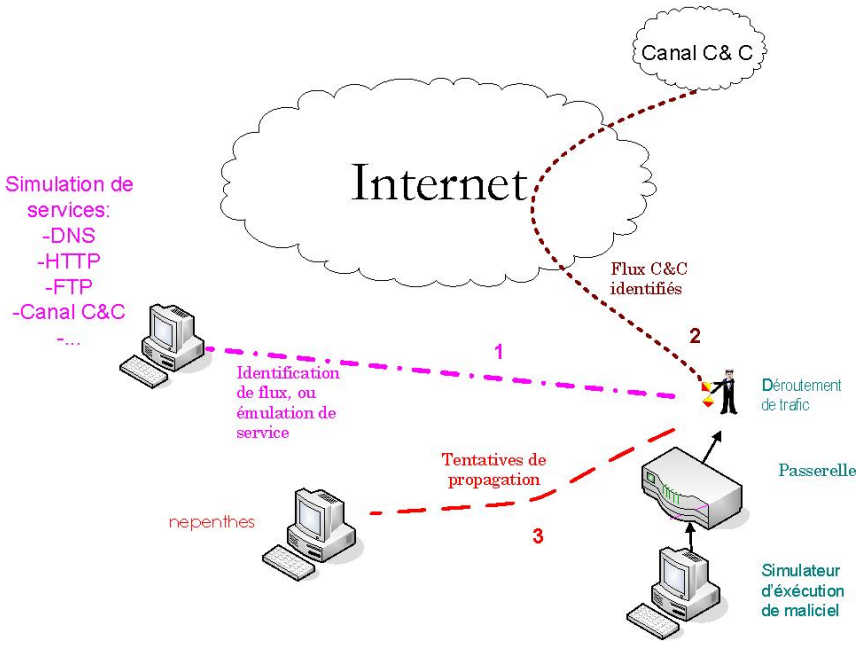

FIG. 2: Environnement d'exécution

Ce système est basé sur la redirection tantôt statique, tantôt dynamique de flux (comme nous l'avions expliqué dans la section précédente, certains flux que nous ne pouvons légitimement autoriser sont redirigés vers une machine de notre réseau local). La redirection statique est implémentée à l'aide d'une fonctionnalité intégrée au pare feu[***]. Cette redirection n'est cependant pas suffisante. En effet les maliciels passent une partie non négligeable de leur temps à scanner Internet dans le but de corrompre d'autres machines. En utilisant une redirection statique nous simulons le fait que la machine réussisse à corrompre toutes les machines de la plage demandée ou aucune, ce qui n'est pas réaliste. Nous avons donc decidé de rediriger dynamiquement une partie seulement de ces adresses, ce qui est beaucoup plus vraisemblable. Les mécanismes mis en jeu sont plus complexes et ne peuvent être expliqués en détails dans ce papier. Nous préférons rediriger le lecteur intéressé vers [ÉAAO+07] pour plus de précisions. Nous avons choisi de ne rediriger qu'une seule adresse par scan pour ces expérimentations.

Enfin pour la simulation des services, nous avons utilisé les logiciels tels que `bind9` pour le service DNS ou `nc` pour simuler les premiers échanges des autres services. Le fait de simuler le service DNS est essentiel. Lorsque l'URL demandée par le maliciel n'est plus disponible, nous pouvons grâce à cette

[***]Destination Network Address Translation (DNAT).

fonctionnalité simuler son existence et ainsi aller proposer un niveau d'interaction supérieur, ce que nous n'aurions pas pu faire avec des redirections à des niveaux protocolaires plus bas.

Pour illustrer le fonctionnement de ce système, l'exemple suivant - correspondant au cas où le maliciel exécuté a pour objectif de scanner le réseau - détaille toutes les étapes de la simulation. Ce comportement qui était théorique au début a été confirmé par la suite au cours de la mise en service de la plateforme.

Dans un premier temps, la passerelle bloque tous les flux en sortie, mis à part les requêtes DNS qui sont redirigées vers un serveur DNS que nous maîtrisons. Le maliciel envoie alors une requête DNS pour obtenir l'adresse IP de l'URL1. Supposons que URL1 existe[†††]. Une fois cette adresse obtenue, le maliciel demande l'ouverture d'une connexion TCP (envoi d'un paquet TCP SYN) vers le port p1 de la machine hébergeant l'URL1. Notre système va alors rediriger cette requête vers une de nos machines locales (flux 1 de la figure 2). Cette machine va accepter l'ouverture de la connexion TCP. Dès lors, comme la plupart du temps le canal C&C du maliciel est de type IRC, les premières requêtes envoyées après l'établissement de la connexion sont du type :

```
USER user1
NICK nick1
```

Cela nous renseigne également sur la localisation du canal C&C du maliciel qui se trouve à (URL1,port1). Nous relançons alors le maliciel en laissant cette fois passer le flux à destination de (URL1,port1) (flux 2 de la figure 2). Grâce à ces échanges sur le canal IRC C&C, notre maliciel connait la tâche qu'il doit effectuer : une tâche de scanning d'autres machines de l'Internet (puisque c'est l'exemple que nous considérons). Le maliciel envoie donc des paquets SYN sur des adresses IP croissantes vers un port donné - le port 445 par exemple. Notre système, qui observe les communications, en déduit qu'il s'agit d'un scan réseau. Le maliciel est alors relancé après avoir reconfiguré le système de redirection dynamique pour qu'une des adresses du scan soit redirigée vers une instance de *nepenthes*. On simule ainsi la réussite de propagation du maliciel (flux 3 de la figure 2).

5 Expérimentation

Nous avons collectés des maliciels pendant 4 mois, ce qui nous a permis de télecharger 46 binaires ayant des empreintes MD5 différentes.

Nous allons à présent décrire les résultats d'analyse les plus significatifs de ces différents binaires.

5.1 Informations sur les canaux C&C

Tous les maliciels tentant de se connecter à un `bot` ont utilisé un protocole similaire à IRC comme canal. Parmi les `botnets` (actifs ou inactifs), nous avons relevé (en observant que les premiers octets envoyés etaient USER <nom_utilisateur> et NICK <surnom> sur ces ports) que les serveurs contactés se trouvaient sur : le port tcp standard, 6667, mais aussi sur les ports tcp 7000,3211,4747,8885,8080 et 22. [CJM05] affirmait que des serveurs pouvaient se trouver sur des ports supérieurs au port standard. Nous avons observé que certains se trouvent sur des ports inférieurs, et en particulier sur des ports inférieurs à 1024 aussi. Il est dès lors nécessaire de considérer que les serveurs IRC utilisés pour implémenter un type de canal C&C peuvent se trouver sur n'importe quels ports. Ces résultats permettent, entre autre, de guider les techniques de détection de botnets présentées au début de l'article en les configurant pour qu'elles regardent d'autres ports que les ports standard.

Nous avons aussi observé, à l'aide de la simulation de réussite d'attaque, que les succès de propagations étaient communiqués au canal C&C.

Après une connexion au serveur et canal IRC, le serveur envoie un ordre de propagation :

```
irc>:$url_serveur_irc$ 332 BiTch|76914 #bitch :
irc>+advscan MassAsn 50 5 0 201.x.x.x -r
```

Le bot informe le canal C&C du début de la tentative de propagation :

[†††] Si URL1 est périmée, on change la configuration de notre serveur DNS pour qu'il désigne une de nos machines comme ayant ce nom.

```
irc>PRIVMSG #bitch :[SCAN]: Random Port Scan started on
irc>201.x.x.x:445 with
irc>a delay of 5 seconds for 0 minutes using 50 threads.
```

Après la simulation d'exécution le succès de propagation est envoyé au canal C&C :

```
irc>PRIVMSG #bitch :[TFTP]: File transfer started to IP:
irc>$@victime_simulee$ ($chemin_vers_executable$).
irc>PRIVMSG #bitch :[TFTP]: File transfered to IP:
irc>$@victime_simulee$($chemin_executable$).
```

Le pirate (c'est-a-dire l'être humain qui contrôle le botnet) peut donc savoir si une machine faisant partie du botnet a corrompu une autre machine ou non. En effet, à la fois la machine attaquante et la machine attaquée (qui par conséquent devient un bot à son tour) communiquent avec le pirate sur les canaux C&C. Ce mécanisme pose des problèmes lorsque l'on essaie d'infilter un botnet avec un pot de miel. En effet, puisqu'un pot de miel n'a pas légalement le droit de lancer des attaques, le pirate va constater que ce (supposé) bot n'essaie pas de se propager et donc qu'il s'agit probablement d'un pot de miel. Et même dans le cas où le pot de miel ment en prétendant sur le canal C&C qu'il a corrompu une autre machine, le pirate pourra soupçonner quelque-chose de louche puisque la machine (supposée) corrompue ne communiquera jamais sur le canal C&C[‡‡‡] pour indiquer qu'elle joint de botnet. Ces problèmes sont soulevés dans [CC06]. Des expérimentations plus poussées sont donc nécéssaires pour savoir si les pirates effectuent effectivement ces tests dans le but de détecter la présence de pots de miel. Dans tous les cas, le renfort de la couverture d'un pot de miel infiltré est un plan de recherche important pour que son infiltration soit solide et que le risque que ce pot de miel espion soit découvert soit limité.

5.2 *Informations sur la structure des* `botnets`

Notre expérimentation nous a montré que les `botnets` n'étaient pas statiques. Un des maliciels, après s'être connecté à un premier canal C&C, a reçu l'ordre de se mettre à jour :

```
irc>:STA 332 Bot|2153 #server# :
irc>.dl http://url/virus.exe> <virus2.exe> 1
```

La première version de ce maliciel avait été détectée par un antivirus à jour. Cependant seule une mise à jour ultérieure nous a permis de détecter sa nouvelle version, l'antivirus ne le détectant pas le jour où on l'avait enregistré.

Après avoir exécuté ce deuxième virus, on constate que ce dernier joint le même serveur IRC mais sur un canal différent, d'où il reçoit l'ordre de propagation :

```
irc>:STA 332 FRA|230440991 #.to.:.asc dcom135 200 0 0 -r -s
```

Nous pouvons en déduire qu'il est fondamental d'analyser en profondeur les flux des canaux C&C pour comprendre la structure complète des botnets. Ainsi, pour tenter de détruire un botnet, la bonne solution n'est probablement pas de désactiver une partie d' un canal C&C dès qu'elle est découverte, mais plutôt d'analyser les flux de ce canal de façon à en déduire d'autres informations intéressantes. Dans l'exemple ci-dessus, l'analyse de `virus2.exe` nous a permis de découvrir l'existence d'autres parties du canal C&C de ce botnet. Ce type d'analyse doit nous permettre de cartographier entièrement un botnet de façon à pouvoir ensuite espérer le détruire efficacement.

6 Conclusion

Notre simulation du comportement de maliciel à un niveau d'intéraction supérieur, tout en n'envoyant pas de trafic malicieux sur Internet, nous a permis d'obtenir des informations quant à la structure des canaux C&C et de la structure plus générales des `botnets`.

Ces résultats encourageants nous poussent à améliorer cette approche selon trois axes différents.

[‡‡‡] Et nous ne pouvons, pour des raisons techniques, pas spoofer sur du TCP, ce qui nous impose irrémédiablement cette contrainte.

– Premièrement lancer ce système plus longuement : les risques liés à ce type d'expérimentations nous ont contraint à ne les lancer qu'en en surveillant en temps réel leur état d'avancement. La confiance obtenue nous amène maintenant à lancer ce type d'expérimentations plus longuement et sans en suivre continuellement le déroulement.
– La deuxième approche consiste à automatiser l'identification des flux. Cette tâche reste cependant encore ardue, les IDS ne faisant que du mieux qu'ils peuvent et n'offrant aucune garantie sur des attaques de type *0day*[§§§] ou sur des attaques connues mais envoyées d'une façon originale [Bid06].
– Le troisième axe de recherche consiste à améliorer la couverture de nos pots de miel infiltrés pour qu'il ne soient pas détectés par les pirates et gardent donc tout leur pouvoir informatif. Nous pensons que la mise en place d'un réseau de pots de miel qui collaborent entre eux permettrait de mieux brouiller les pistes pour le pirate. Nous travaillons donc à la conception (théorique pour le moment) des mécanismes de collaboration au sein d'un tel réseau de pots de miel, et essayons d'en évaluer la performance en termes de couverture anti détection et de qualité (et quantité) des informations récupérées sur les botnets et leurs pirates.

D'autres axes d'amélioration de notre approache d'étude des botnets seront également développés dans un futur relativement proche. Toutefois, ce travail d'infiltration des botnets a pour objectif final de pouvoir mettre en place à l'échelle de l'Internet un système de protection proactif global. Ainsi, nous espérons que nos pots de miel infiltrés recevront des ordres de lancement d'attaques, et qu'il sera donc aisé de bloquer les attaques qui suivront, voire mieux, de bloquer la diffusion des ordres d'attaque. A plus courte échéance, ce travail d'infiltration nous procure une liste de machines corrompues et pour lesquelles le risque de participation à une attaque est élevé. Cette information est intéressante car elle aide la gestion des politiques de sécurité. Par exemple, pour un IDS, les seuils de détection d'attaques peuvent légitimement être renforcés pour ces machines identifiées comme corrompues, la probabilité qu'une anomalie du profil du trafic généré par ces machines corresponde à une attaque (plutôt qu'une augmentation légitime du trafic) étant sensiblement plus élevée.

Références

[AKS+06] Mitsuaki Akiyama, Takanori Kawamoto, Masayoshi Shimamura, Teruaki Yokoyama, and Suguru Yamaguchi. A proposal of metrics for botnet detection based on its cooperative behavior. In *Worskshop SAINT 2007*. IEEE, 2006.

[Bid06] Renaud Bidou. Contournement des ids/ips pour les nuls. *Actes SSTIC*, 2006.

[BKH+06] Paul Baecher, Markus Koetter, Thorsten Holz, Maximillian Dornseif, and Felix C. Freiling. The nepenthes platform : An efficient approach to collect malware. In *RAID*, pages 165–184, 2006.

[CC06] Cliff C.Zou and Ryan Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *Proceedings of the 2006 International Conference on Dependable Systems and Networks (DSN'06)*, Orlando,FL 32816-2362, 2006.

[CJM05] Evan Coke, Farnam Jahanian, and Dany McPherson. The zombie roundup : Understanding, detecting, and disrupting botnets. In *Sruti'2005*, pages 39–44. USENIX, 2005.

[FHW05] Felix Freiling, Thorsten Holz, and Georg Wicherski. Botnet tracking : Exploring a root-cause methodology to prevent distributed denial-of-service attacks. Technical Report AIB-2005-07, RWTH Aachen, 2005.

[Hol05] Thorsten Holz. A short visit to the bot zoo. *IEEE Security & Privacy Magazine*, 3 :76–79, May-June 2005.

[HP05] Thorsten Holz and Fabien Pouget. A pointillist approach for comparing honeypots. In *Proceedings of the 'Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA 2005)' Conference, Vienna, 7.-8. July 2005*, volume 3548 of *Lecture Notes in Computer Science*, pages 51–68, Frankfurt am Main, Germany, 2005. Spriger.

[§§§] Outil d'exploitation d'une vulnérabilité pour laquelle aucun correctif n'a été publié.

[Kri04] John Kristoff. Botnets. In *32nd Meeting of the North Americain Network Operators Group*, October 2004.

[Mas06] Yuri Mashevsky. Les dessous de l'économie souterraine des codes malicieux : chevaux de troie, virus et malware. *VirusList.com*, 2006.

[MMS01] David More, Geoffrey M.voelker, and Stefan Savage. Inferring internet denial-of-service activity. In *Proceedings of the 10th USENIX Security Symposium, Washington, D.C, USA*. The USENIX Association, 2001.

[Rac04] Stéphane Racine. Analysis of internet relay chat usage by ddos zombies. Master's thesis, April 2004.

[Sym06] Symantec. Symantec internet security threat report, September 2006. `http://www.symantec.com/enterprise/threatreport/index.jsp`.

[ZP05] Guangsen Zhang and Manish Parashar. Cooperative mechanism against ddos attacks. In *International Conference on Security Management (SAM .05), Las Vegas, NV, USA*. The Applied Software System Laboratory, Department of Electrical and Computer Engineering, Rutgers University, CSREA Press, 2005.

[ÉAAO+07] Éric Alata, Ion Alberdi, Philippe Owezarski, Vincent Nicomette, and Mohammed Kâaniche. Mécanismes d'observations d'attaques sur internet avec rebonds. *A paraître dans les Actes SSTIC*, 2007.