Observation et analyse d'attaques sur Internet

Ivan Studnia

Master Recherche en Informatique 2010-2011

Rapport de stage

Observation et analyse d'attaques sur Internet

Auteur :
Ivan Studnia
<ivan.studnia@telecom-bretagne.eu>

Encadrants :
Éric Alata
Mohamed Kaâniche
Vincent Nicomette
(Groupe TSF)

Conseiller d'études :
Christophe Lohr

Avril à Septembre 2011

Résumé

Dans le domaine de la sécurité informatique, un pot de miel est un outil dont le but est d'attirer les attaquants et d'enregistrer leurs activités, afin de mieux connaître leurs techniques et objectifs. Dans cette optique, un pot de miel a été conçu et déployé au LAAS depuis 2006 afin d'observer le trafic malveillant utilisant le protocole `ssh` sur Internet. De nombreuses analyses ont été réalisées et concernent exclusivement ce site géographique. Pour compléter et enrichir ces analyses, de nouveaux déploiements doivent être réalisés en plusieurs sites géographiques. L'objectif est de comparer les tendances relevées en ces différents lieux avec celles ayant été observées précédemment au LAAS. Ce rapport présente un état de l'art des pots de miel, la conception de l'infrastructure déployée, les différents programmes développés pour superviser cette infrastructure et guider les analyses. Par la suite, les premières données collectées sont détaillées et analysées globalement et en considérant différents critères tels que la source géographique des attaques et le type d'attaque.

Mots-clés : *sécurité informatique, Internet, attaques par dictionnaire, intrusions, comportement d'attaquants*

Abstract

Honeypots are tools designed to lure attackers which record their activities so we can have a better understanding of their goals and their techniques. Therefore, a honeypot has been set up at LAAS in 2006 to watch over malevolent traffic on the Internet through `ssh`. Many analyses have been performed on these data, however restricted to that one place. In order to improve these analyses, new honeypots have to be installed in other locations. Our goal is to compare the newly recorded trends with those previously seen at LAAS. This report begins with a state of the art survey of honeypots then exposes the main goals of our works. The structure of our system is then explained, along with the many programs we made to manage this and prepare our analyses. Subsequently, the first data recorded are showed and analysed as a whole but also according to various criteria, like the origin or type of the attacks.

Keywords: *computer security, Internet, dictionary attacks, intrusions, attackers' behaviour*

Remerciements

Je tiens à remercier dans un premier temps toute l'équipe pédagogique de Telecom Bretagne et les intervenants du parcours Systèmes Informatiques Centrés sur l'Humain de la formation du Master 2 Recherche en Informatique de l'Université de Rennes 1.

Je remercie également Monsieur Christophe LOHR pour ses conseils donnés lors de son suivi de ce stage.

Je tiens à remercier tout particulièrement et à témoigner toute ma reconnaissance aux personnes suivantes, pour l'expérience que j'ai vécue durant ces six mois au sein du Laboratoire d'Analyse et d'Architecture des Systèmes :

Madame Karama KANOUN, Responsable du Groupe de recherche Tolérance aux fautes et Sûreté de Fonctionnement informatique, pour son accueil et la confiance qu'elle m'a accordée dès mon arrivée.

Messieurs Éric ALATA, Mohamed KAÂNICHE et Vincent NICOMETTE, mes responsables de stage, pour m'avoir accordé toute leur confiance, pour le temps qu'ils m'ont consacré tout au long de cette période ainsi que pour leurs conseils lors de la rédaction de ce rapport.

Enfin, je remercie tous les stagiaires (et ex stagiaires) et doctorants du groupe TSF pour leur bonne humeur et la bonne ambiance omniprésente qui règne dans ces locaux.

Table des matières

Chapitre 1

Introduction

Le développement très rapide d'Internet (près de deux milliards d'utilisateurs recensés en 2010 selon l'Internet World Stats [1]) a fait apparaître de nombreux services accessibles en ligne, ainsi que de nombreuses communautés d'internautes. L'importance prise par ce réseau est telle que des règles et même des lois ont été créées afin d'assurer son bon fonctionnement.

En effet, certains utilisateurs détournent l'outil informatique à des fins malveillantes en exploitant des failles matérielles ou logicielles. Ces pirates informatiques vont ainsi pouvoir accéder à des données confidentielles, prendre le contrôle de machines ou répandre des logiciels malveillants (vers, chevaux de Troie...) en dépit des lois. Par conséquent, des contre mesures ont été développées pour tenter de combler ces failles et contrer les attaques, obligeant continuellement les pirates à rechercher de nouvelles vulnérabilités et imaginer de nouvelles attaques pour les exploiter, lançant attaquants et experts en sécurité dans une perpétuelle course de type attaque-défense[1].

Il est donc aujourd'hui primordial de connaître les stratégies utilisées actuellement par les pirates afin de parer au mieux les prochaines attaques et d'élaborer des nouveaux mécanismes de protection adaptés. Il faut pour cela collecter le plus d'information possible sur les pirates informatiques. Un certain nombre de techniques sont aujourd'hui employées dans cet objectif

1. Les initiatives de centralisation de données recueillies par des sondes qui récupèrent les données fournies par divers équipements comme des routeurs ou des pare-feux [2].
2. La mise en place de « leurres », d'abord manuellement[2], puis de manière automatique, afin de piéger un pirate pour pouvoir l'observer.

C'est l'automatisation de ce dernier aspect qui va nous intéresser ici, à travers la notion de « pot de miel ».Un pot de miel (*honeypot*) est un système informatique connecté à un réseau sur lequel ont volontairement été laissées certaines vulnérabilités dans le but d'attirer les attaques, de collecter puis d'analyser des informations sur leurs actions (protocoles employés, failles exploitées, programmes utilisés...). Le but de ce stage sera de poursuivre les analyses faites sur un pot de miel déployé au LAAS et de contribuer à l'amélioration de celui-ci, en cherchant notamment à voir si les comportements des attaquants varient selon l'emplacement des machines qu'ils attaquent. Cela implique donc une modification de l'architecture existante afin de permettre un déploiement du pot de miel sur différents sites géographiques et la récupération des données enregistrées.

Dans ce rapport, la première partie consiste une brève présentation du laboratoire qui m'a accueilli pour ce stage, le LAAS.

La deuxième partie fait un état de l'art sur les différents types de pots de miel utilisés, en s'intéressant à leurs objectifs et leurs caractéristiques. Nous nous penchons ensuite plus en détail sur le pot de miel ayant été déployé au LAAS, qui servira de base pour l'élaboration de nos

1. http ://www.internetworldstats.com/stats.htm
2. www.dshield.org

nouvelles expériences, ainsi que sur les premiers résultats obtenus lors des années précédentes. Enfin, nous y posons les bases du travail effectué lors de ce stage.

Dans une troisième partie, nous détaillons le fonctionnement du nouveau système de pots miel que nous avons conçu durant ce stage, en nous focalisant sur les causes qui ont mené à ce choix d'architecture.

La quatrième partie détaille notre protocole expérimental, regroupe et commente les résultats de diverses analyses qui ont été faites sur les données collectées lors de ces expériences et présente nos travaux en cours.

Chapitre 2

Présentation du laboratoire

Ce stage a été réalisé dans l'équipe TSF qui fait partie de SINC, un des quatre pôles du LAAS-CNRS. Nous présentons ci-dessous ces différentes entités.

2.1 Le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)

Créé en 1968, le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) comporte aujourd'hui 640 personnes : 86 chercheurs CNRS, 120 professeurs et maîtres de conférence, 115 ingénieurs, techniciens et administratifs, 273 doctorants et 46 post-doc.

Les recherches, en sciences et technologies de l'information, de la communication et des systèmes, sont conduites au sein de 19 groupes de recherche, actifs dans 4 domaines : micro et nano systèmes, automatique et traitement du signal, informatique, et robotique.

Au sein de ces 19 groupes, le LAAS mène des recherches en sciences et technologies de l'information, de la communication et des systèmes dans quatre grands pôles :

- les micro et nano systèmes et technologies (MINAS),
- l'optimisation, la commande et le traitement du signal (MOCOSY),
- les systèmes informatiques critiques (SINC),
- la robotique et l'intelligence artificielle (RIA).

Le LAAS est un des grands laboratoires de recherche du CNRS, également associé à l'Université de Toulouse. Des recherches multidisciplinaires y sont conduites selon deux axes transverses : les interactions avec le vivant et l'intelligence ambiante.

2.2 Pôle de Recherche Systèmes INformatiques Critiques (SINC)

Les thèmes de recherche développés au sein du pôle Systèmes INformatiques Critiques (SINC) concernent la conception et l'analyse de systèmes de traitement de l'information complexes qui doivent satisfaire des propriétés imposées par des applications fortement contraintes. Ces propriétés s'expriment par un ensemble d'exigences, définies en termes de contraintes temporelles, de qualité de service, de sûreté de fonctionnement, de sécurité informatique et d'actions coopératives. Ainsi, les principaux travaux concernent :

- L'ingénierie des systèmes complexes mettant l'accent sur les processus de conception et d'évaluation.
- La formalisation des étapes de spécification, de vérification et de test des logiciels, en renforçant notamment l'intégration des outils associés au processus de développement industriel.
- La qualité de service des architectures réparties, des protocoles et réseaux de communication pour des applications coopératives et de calcul intensif.

- La sûreté de fonctionnement et la résilience des systèmes informatiques vis-à-vis de différentes aspects : interactions entre les composants, mobilité des utilisateurs, des dispositifs et des services, etc.
- L'évaluation quantitative des propriétés par la modélisation stochastique, par de la métrologie en environnements réels ou simulés, en fonctionnement nominal ou perturbé, et par l'analyse et la caractérisation des comportements ainsi obtenus.
- La définition de politiques et de modèles, ainsi que le développement de nouvelles technologies, respectant à la fois les exigences de sécurité et de protection de la vie privée.

2.3 Groupe de Recherche Tolérance aux fautes et Sûreté de Fonctionnement informatique (TSF)

Les travaux du groupe TSF du LAAS-CNRS portent sur la sûreté de fonctionnement des systèmes informatiques, définie comme l'aptitude d'un système à délivrer un service de confiance justifiée. Elle englobe les propriétés de disponibilité, fiabilité, intégrité, confidentialité, maintenabilité, sécurité par rapport aux défaillances catastrophiques (sécurité-innocuité), ainsi que de sécurité vis-à-vis des malveillances (sécurité-immunité).

Les recherches concernent à la fois la conception, le test et l'évaluation. Une autre caractéristique forte concerne l'étendue des fautes prises en compte : les fautes physiques du matériel, les fautes du logiciel et les fautes d'interaction malveillantes, c'est-à-dire les intrusions. De plus, le groupe s'appuie depuis de nombreuses années sur le développement conjoint de méthodes analytiques et expérimentales. Tout cela confère au groupe un caractère tout à fait saillant dans le domaine de la sûreté de fonctionnement des systèmes informatiques.

Les recherches conduites couvrent quatre volets :
- la prévention des fautes, qui vise à empêcher l'occurrence ou l'introduction de fautes ;
- la tolérance aux fautes, destinée à assurer la fonction d'un système en dépit des fautes ;
- l'élimination des fautes, dont l'objet est de réduire le nombre ou la sévérité des fautes ;
- la prévision des fautes, afin d'estimer la création, la présence et les conséquences des fautes.

Une partie des travaux effectués à TSF concerne ainsi la sécurité des systèmes informatiques contre les malveillances, matérielles ou logicielles. Cet axe de recherche a en partie débouché sur des travaux concernant les attaques lancées contre des systèmes par Internet et en particulier sur leur étude via des pots de miel. Ce stage utilise ainsi les outils et résultats produits par ces précédents travaux afin d'étendre le champ de ces études.

Chapitre 3

État de l'art et contexte des travaux

Dans cette partie, nous explicitons le concept de pot de miel en détaillant plusieurs approches. Comme mentionné précédemment, un pot de miel est un système (réel ou virtuel) connecté à un réseau sur lequel on a volontairement laissé des vulnérabilités simples à exploiter (failles) dans le but d'attirer les attaquants. Le concept du pot de miel se décline de nombreuses façons, selon les objectifs que l'on cherche à atteindre ou les informations que l'on souhaite obtenir. Nous allons ainsi présenter certaines de ces catégories, en commençant par celles dites historiques, et les illustrer avec des exemples concrets. Par suite nous présenterons plus en détail le pot de miel développé au LAAS ainsi que quelques résultats précédemment obtenus via l'analyse des données collectées par celui-ci, puis nous conclurons ce chapitre en présentant les objectifs de nos travaux actuels.

3.1 Classification des pots de miel

Historiquement, les pots de miel pouvaient être classés en deux catégories, haute ou basse interaction, selon le niveau d'interaction qu'ils proposaient aux pirates. Cependant, de nouvelles tendances font leur apparition, tels les pots de miel dits hybrides, intermédiaires entre les deux précédentes, ou encore des pots de miels adaptatifs. Cette partie décrira donc tout d'abord les deux catégories historiques, puis s'intéressera à quelques exemples des nouvelles pistes explorées. Au vu du grand nombre d'implémentations existantes de pots de miel, cette étude reste partielle en se focalisant sur quelques exemples, mais de plus amples informations sont par exemple disponibles dans [3], [4] et [5].

3.1.1 Basse interaction

Principe

Un pot de miel basse interaction, comme son nom l'indique, offre très peu d'interaction à l'attaquant. Très souvent, cela consiste à ne pas fournir à l'attaquant de vrais services mais des émulations de service au travers de simples interfaces très basiques. L'attaquant ne peut donc interagir avec le pot de miel qu'au travers de ces services et ses actions entraîneront uniquement les réponses prévues lors de la configuration du pot de miel. Les possibilités de l'attaquant sont donc restreintes, mais on a la quasi certitude que celui-ci ne pourra pas accéder au système hébergeant le pot de miel ni compromettre d'autres systèmes.

Les pots de miel basse interaction ont l'avantage d'être aisés à mettre en place et à contrôler (pas de danger lié aux attaques) tout en permettant d'obtenir des informations intéressantes sur le trafic malveillant. Cependant, les données collectées concernent des dialogues relativement courts entre l'attaquant et le système. En effet, à cause du faible niveau d'interaction proposé,

l'attaquant ne pourra réaliser que très peu d'actions sur le système et risque même de se rendre compte qu'il a affaire à un pot de miel, ce qui va influer sur son comportement (et souvent provoquer une déconnexion de sa part).

Les pots de miel basse interaction sont donc surtout utiles pour étudier l'évolution ou l'intensité d'attaques (dans le cas de la propagation d'un ver par exemple) [6].

Exemple : Honeyd

`Honeyd` [7] est un programme permettant de créer plusieurs pots de miel virtuels sur une machine. Il peut simuler les protocoles TCP et UDP et répond aux messages ICMP. D'après la figure 3.1, on peut voir que lorsque les paquets entrants contiennent une requête de connexion, une application simulant le service demandé (`ftp, ssh, etc.`) peut être lancée. Les paquets sortants sont modifiés par le *personality engine* afin de respecter l'empilement réseau du système d'exploitation que l'on souhaite simuler. La partie routage n'intervient que lorsqu'on veut modéliser un réseau entier avec honeyd.

`Honeyd` peut instancier plusieurs pots de miel virtuels sur un seul système physique et simuler divers systèmes d'exploitation en émulant leur pile réseau, tout en gardant une cohérence vis à vis de ceux-ci (pas de ssh sur Windows par exemple).

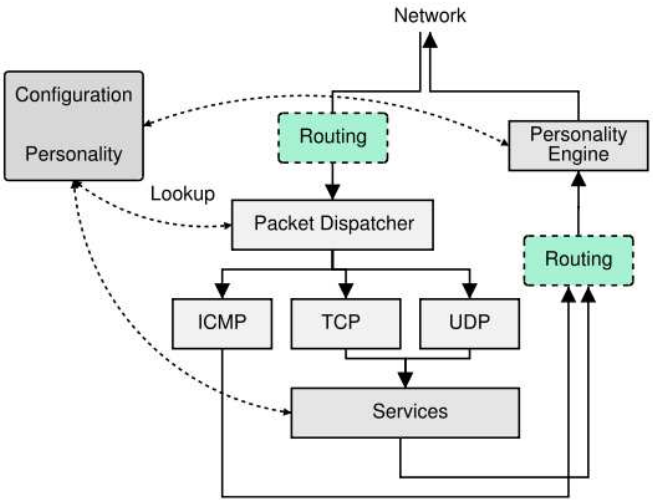

FIGURE 3.1 – (Figure 2 de [7]) Architecture d'un pot de miel `Honeyd`

Comme vu dans [5] et [6], la facilité de déploiement et de configuration de pots de miel basse interaction a conduit à l'apparition de projets visant à regrouper les données fournies par des pots de miel identiquement configurés déployés en de multiples endroits. Leurre.com, par exemple, est basé sur `Honeyd` et dispose de pots de miel installés dans une trentaine de pays et sur quatre continents [1].

3.1.2 Haute interaction

Principe

Un pot de miel haute interaction correspond à un système réel ou une émulation d'un système complet. Les services qui sont offerts à l'attaquant sont donc de réels services avec lesquels il peut totalement interagir et donc exploiter. Celui-ci peut alors pénétrer jusqu'au coeur du système.

1. www.leurrecom.org/project.php

L'objectif est ainsi d'observer des mécanismes d'attaque plus poussés. Le contrecoup de cette méthode est qu'en laissant plus de libertés à l'attaquant, le risque que celui-ci puisse nuire devient réel. Il faut s'assurer que les actions de l'attaquant n'affectent pas d'autres systèmes. En effet, il pourrait vouloir utiliser le système pour attaquer d'autres machines présentes sur le réseau (ou via Internet), ce qui doit être évité à tout prix. Par conséquent, une première problématique réside dans le fait de trouver un bon équilibre entre interaction et sécurité.

Par ailleurs, l'autre problème qui se pose est celui de la collecte des données. Celle-ci doit se faire de manière suffisamment discrète pour que l'attaquant ne se doute pas qu'il est observé (par exemple il ne faut pas qu'il puisse voir un processus correspondant, ou que la collecte de données ralentisse fortement l'exécution des commandes), mais elle doit également être assez robuste au cas où celui-ci ne voudrait pas laisser de traces de son passage (et effacerait donc un certain nombre de fichiers). Bien qu'un pot de miel haute interaction puisse être implémenté en tant que système physique, les contraintes en terme de coûts et de maintenance font de la virtualisation la méthode la plus employée pour déployer ces pots de miel.

Exemple : Sebek

`Sebek` [8] est un pot de miel physique et base sa récupération des données sur une communication client-serveur (voir figure 3.2). Chaque client correspond à un pot de miel dans le noyau duquel est installé un module qui intercepte des appels système. Cela va permettre de remplacer l'exécution de l'appel standard par la version de `Sebek`, qui va appeler l'original en plus d'enregistrer les données qui lui étaient destinées.

`Sebek` peut ainsi intercepter plusieurs appels système (`read()` par exemple), ce qui permet d'obtenir des informations envoyées par l'attaquant une fois qu'elles ont été déchiffrées (si l'attaquant chiffre ses transactions) mais avant qu'elles ne soient utilisées par le système. Ce faisant, il envoie cette information au serveur via le réseau, mais cache ces paquets à tous les pots de miel grâce à une autre modification du noyau concernant la gestion des protocoles réseau. Ainsi, l'attaquant ne pourra pas voir ces paquets, même en ayant un analyseur de réseau installé sur un système utilisant `Sebek`.

En théorie, `Sebek` permet de collecter de nombreuses informations sur l'attaquant et ce de manière relativement discrète. Cependant, `Sebek` et ses techniques de dissimulation sont désormais bien connus des attaquants et ceux-ci ont développé des contre mesures pour le détecter ou l'inhiber ([9], [1]).

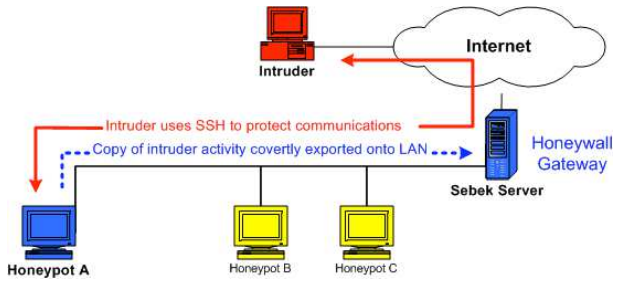

FIGURE 3.2 – (Figure 1 de [8]) Implémentation typique de `Sebek`

3.1.3 Nouvelles tendances

Pot de miel hybride

Dans [6], il est montré que dans le cadre d'un déploiement massif, pots de miel haute et basse interaction peuvent se compléter efficacement. En effet, un pot de miel haute interaction reçoit bien (40 fois d'après [6]) plus de paquets qu'un pot de miel basse interaction. Cependant, sur des installations similaires (même nombre de pots haute et basse interaction, mêmes ports ouverts, etc.) les types de requêtes adressés à des pots de miel haute et basse interaction sont du même ordre. La différence de quantité d'information collectée vient donc surtout du fait que les transactions avec les pots de miel basse interaction sont interrompues prématurément faute de réponses adaptées aux requêtes demandées. Ainsi, il serait intéressant d'utiliser les informations supplémentaires fournies par un pot de miel haute interaction afin de mieux configurer des pots basse interaction pour tenter d'acquérir plus de données et ce à un moindre coût.

C'est sur cette idée qu'est par exemple basé `Scriptgen` [10]. `Scriptgen` est un programme dont le but est de générer automatiquement des scripts permettant d'émuler des protocoles pour `Honeyd` (décrit dans 3.1.1) à partir d'un échantillon de données collectées grâce à tcdpdump sur des pots de miel haute interaction et ce sans avoir de connaissance des protocoles *a priori*.
`Scriptgen` fonctionne en quatre temps :

- Tout d'abord, un module va analyser les trames de l'échantillon afin de recomposer les séquences d'échanges client-serveur correspondantes.
- Ensuite, ces séquences vont permettre de construire une première machine à états grossière dont les différents parcours correspondront à autant de séquences. Chaque transition correspond à une requête du client et est pondérée par sa fréquence d'apparition. Chaque état correspond à un ensemble de réponses possibles de la part du serveur. Pour limiter la complexité, le nombre maximum d'états ainsi que le nombre maximum de transitions sortant d'un état sont fixés à l'avance.
- Cet automate va ensuite être simplifié afin de déterminer des échanges client-serveur types grâce à deux techniques de *clustering* différentes. La première, dite de *macro-clustering*, regroupera certaines transitions sortantes entre elles en fonction de leur similarité, alors que la suivante, dite de *micro-clustering* travaillera sur ces nouveaux regroupements et en partitionnera certains afin d'éviter une trop forte simplification de l'automate. Le *macro-clustering* se contente de travailler sur la similarité bit à bit des séquences analysées pour les regrouper puis le *micro-clustering* cherche des chaînes récurrentes parmi les éléments variables de ces regroupements pour affiner leur partitionnement.
- Enfin, une fois cet automate simplifié construit, un script pour `Honeyd` est généré à partir de celui-ci.

Les résultats obtenus avec des pots de miel basse interaction configurés grâce à `Scriptgen` montrent que ceux-ci parviennent à établir une conversation avec le client et simuler un service donné durant un certain temps, cependant inférieur à celui obtenu avec le véritable service s'éxécutant sur un pot de miel haute interaction. Par ailleurs, `Scriptgen` étant extrêmement dépendant de la qualité de l'échantillon utilisé ainsi que des divers seuils utilisés pour la simplification de l'automate, ces paramètres vont influer fortement sur les résultats obtenus. En revanche, le fait de pouvoir simuler des services sans pour autant connaître leur fonctionnement sur des pots de miel basse interaction laisse entrevoir des évolutions très encourageantes pour cette méthode.

Pot de miel adaptatif

Toujours afin de maximiser la quantité d'informations obtenues lors d'une attaque, une autre voie actuellement explorée est celle des pots de miel adaptatifs. Le but est ici de pousser l'attaquant à tenter le plus de techniques possibles pour atteindre son objectif. Pour ce faire, le pot

de miel va par exemple parfois bloquer l'exécution d'une commande [11]. Ainsi, interrompre le téléchargement d'un fichier entrepris par l'attaquant le poussera peut être à essayer de le récupérer sur un nouveau dépôt, dévoilant ainsi son existence. En modélisant l'interaction entre un attaquant et leur pot de miel comme un jeu, Wagener et al. [11] se sont alors basés sur la théorie des jeux ainsi que sur des données fournies par d'autres pots de miel (séquences types de commandes entrées par un attaquant) pour calculer les meilleures probabilités pour leur pot de miel de bloquer l'exécution d'un programme :

- Les parties entre le pot de miel et un attaquant sont représentées par des arbres dont les noeuds sont des commandes pour l'attaquant et les décisions (bloquer ou non l'exécution) pour le pot de miel. Les branches sont pondérées par les probabilités de passer d'un état au suivant.
- Pour l'attaquant, le but du jeu est d'atteindre son objectif en un minimum de transitions.
- Pour le pot de miel, le but est de faire durer la partie en poussant l'attaquant à prendre des chemins plus longs.

Ainsi, en utilisant les données déjà collectées par un pot de miel haute interaction et en opérant un processus d'apprentissage selon certaines règles dessus, le pot de miel adaptatif peut permettre de collecter de nouvelles informations par rapport à un haute interaction « standard ».

Collecteur de maliciels (*malware*)

Si les mécanismes présentés dans la partie 3.1.3 visaient à améliorer l'efficacité d'un pot de miel en présence d'un attaquant humain, `nepenthes` [12] s'intéresse aux outils automatisés. De par sa structure, il se rapproche d'une plate-forme accueillant des pots de miel basse interaction d'un type bien particulier. En effet, l'objectif de `nepenthes` est de télécharger le plus de maliciels possible. Par maliciels, nous entendons des programmes qui infectent des ordinateurs et se répandent d'eux mêmes sur Internet. L'intérêt de cette opération est double. D'une part cela permet d'avoir une connaissance sur la circulation des programmes utilisant des vulnérabilités connues. D'autre part, cela peut aussi permettre de découvrir de nouvelles attaques ciblant des vulnérabilités non publiées pour lesquelles il n'existe aucun correctif disponible (*zero-day attack*). En effet, si un module de `nepenthes` est activé par un de ces nouveaux maliciels, il est possible de détecter qu'il n'utilise pas les failles prévues lors de la configuration de la plate-forme et ne rentre donc pas dans les paramètres de la simulation. Dès lors, les transactions peuvent par exemple être redirigées vers un pot de miel haute interaction qui enregistrera alors les nouvelles informations.

Pour ce faire, `nepenthes` répartit son travail sur différents modules aux tâches bien précises, comme illustré dans la figure 3.3. Les modules simulant des vulnérabilités (*vulnerability modules* vont émuler les parties vulnérables d'un système et vont donc recevoir les données envoyées par un programme attaquant (se rapprochant dans le principe d'un pot de miel basse interaction). Via d'autres modules, `nepenthes` va alors extraire de ces données les informations nécessaires (une URL par exemple) pour pouvoir récupérer le maliciel que l'attaque cherchait à lui faire installer. Une fois les fichiers récupérés, une dernière famille de modules est chargée de les traiter (stockage, transfert à d'autres bases, envoi à des compagnies d'antivirus, etc.).

3.2 Pot de miel du LAAS

3.2.1 Objectifs

Beaucoup de données concernant la propagation et le fonctionnement de nombreux maliciels ont déjà été recueillies grâce aux pots de miel. En revanche, la quantité d'information à propos des attaquants humains est bien plus faible, probablement car plus difficile à obtenir. L'objectif principal des pots de miel déployés au LAAS est ainsi l'observation en détail du comportement de ces

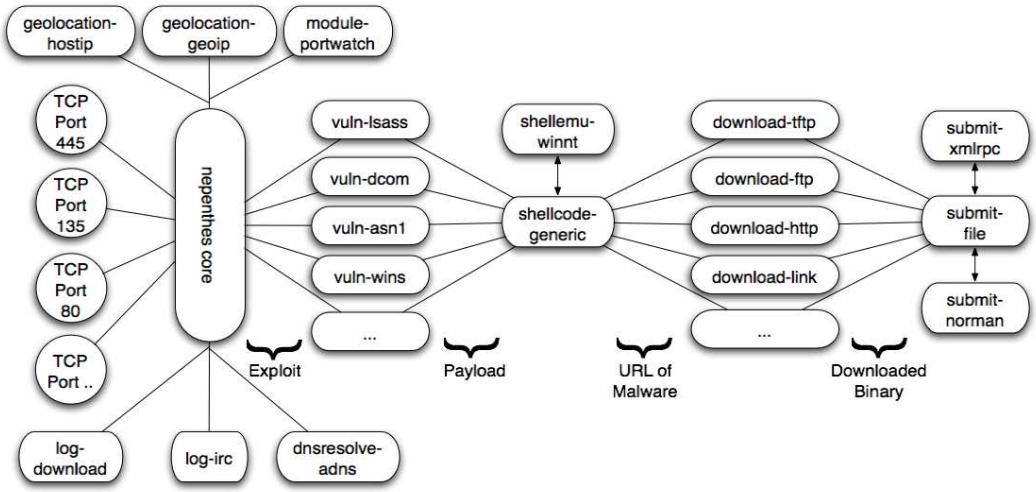

FIGURE 3.3 – (Figure 1 de [12]) Schéma conceptuel de `nepenthes`

attaquants humains. Il doit donc répondre à trois problématiques de base : être « attrayant »pour les attaquants, collecter le plus d'informations possibles sur leurs activités et contrôler ces activités afin d'éviter tout débordement, tout en maintenant une certaine transparence pour ne pas éveiller de soupçons.

Le pot de miel haute interaction permet d'obtenir une interactivité suffisante pour leurrer un attaquant humain sur le long terme afin d'observer son comportement, sa stratégie. Cependant, les attaques sur Internet ne sont pas toutes lancées par des humains, mais également par des outils automatiques, tels que des vers se propageant sur le réseau par exemple. Dans cette partie, nous nous détaillerons le fonctionnement du pot de miel haute interaction développé au LAAS puis nous nous intéresserons aux principaux résultats des analyses effectuées sur les données collectées lors de son déploiement.

Faille retenue Pour attirer au mieux des attaquants humains, la décision a été prise de choisir des vulnérabilités plus facilement exploitables par ceux-ci. Pour ce faire, la vulnérabilité retenue est la création dans un système `GNU/Linux` de comptes utilisateurs ayant des mots de passe simples à deviner (typiquement nom du compte et mot de passe identiques). Ces comptes seront accessibles via le service ssh.

Nature Afin d'obtenir le plus haut niveau d'interaction possible, plusieurs machines doivent être disponibles pour les attaquants. Comme vu dans la partie 2, les systèmes physiques sont plus coûteux et complexes à administrer. Par conséquent, la virtualisation de plusieurs pots de miel sur une seule machine physique permet de réduire ces problèmes.

Observabilité Elle doit être assez haute pour pouvoir retranscrire la complexité de scénarios d'attaques menés par des humains.

Informations recherchées Les informations retenues afin de pouvoir reconstituer les scénarios d'attaques sont :

- Les couples (nom d'utilisateur, mot de passe) tentés par l'attaquant.
- Les caractères tapés par l'attaquant ainsi que ceux qui s'affichent sur son terminal. Cela permet de savoir quelles commandes a entrées l'attaquant.

– Les fichiers exécutés par le système d'exploitation, dans le cas où la liste des caractères ne suffirait pas à déterminer cela (raccourcis claviers ou programme en appelant d'autres par exemple).

3.2.2 Implémentation

Les pots de miel sont des systèmes d'exploitation `GNU/Linux` installés sur des machines virtuelles via `Qemu`, dont les sources ont été modifiées pour permettre d'extraire facilement le contenu d'une partie de la mémoire des machines virtuelles qu'il héberge.

Collecte des données Il existe nombre de solutions possibles pour analyser les paquets qui transitent par un pot de miel. Une des plus simples consiste à utiliser un analyseur de réseau pour récupérer certaines trames. Cependant, il faut pouvoir lire le contenu chiffré qui transite et donc pouvoir récolter les données après leur déchiffrage. La solution retenue ici est ainsi le développement d'un greffon (plus difficilement détectable qu'un module [9]) appliqué au noyau pour y apporter certaines modifications :

– Modification du pilote tty qui fournit les routines de lecture et d'écriture sur le terminal d'un utilisateur distant. En insérant des fonctions de sauvegarde dans ce pilote, on a alors accès au contenu du terminal (d'une manière similaire à celle décrite en [8]).
– Une modification du même ordre est faite sur la routine d'exécution des programmes, afin de récupérer la liste des fichiers exécutés par l'attaquant.
– Enfin, un nouvel appel système est créé qui permet à des programmes sélectionnés (par exemple : ssh) s'éxécutant dans l'espace utilisateur de stocker des données dans l'espace noyau.

Archivage des données La récupération des données ne se fait pas en temps réel, mais à heure fixe, ce qui réduit la surcharge du système et augmente la transparence mais implique un stockage temporaire des données sur le pot de miel. Ce stockage se fait dans une zone statique de la mémoire de l'espace noyau. Cette zone est de taille fixe pour ne pas surcharger les appels systèmes avec de l'allocation dynamique.

Pour limiter l'inconvénient de la taille fixe en cas de surabondance de données, celles-ci sont compressées par l'algorithme de compression de données LZRW1. Si cette compression s'avérait insuffisante, les pertes de données seraient inévitables mais seraient détectées grâce à un indice s'incrémentant à chaque fois qu'une nouvelle information doit être enregistrée et ce même si elle n'est pas archivée faute de place. Ainsi deux données consécutives en mémoire n'ayant pas un indice consécutif dénoteront une perte d'information.

Récupération des données Les pots de miel étant en fait des machines virtuelles s'exécutant sur un système d'exploitation hôte, la mémoire virtuelle est directement accessible par ce dernier. Ainsi, en encadrant la zone mémoire de stockage des systèmes virtuels définie précédemment par une certaine séquence, on peut identifier cette zone dans le système hôte. La récupération des données se fait alors en bloquant temporairement le système invité, en récupérant les informations contenues dans ladite zone. La mémoire est ensuite vidée, puis le système invité est finalement débloqué. La durée de ces opérations est suffisamment rapide (inférieure à une seconde) pour ne pas éveiller de soupçon chez un attaquant qui serait présent à ce moment là.

La figure 3.4 donne un résumé de l'implémentation de la collecte de données dans ce pot de miel.

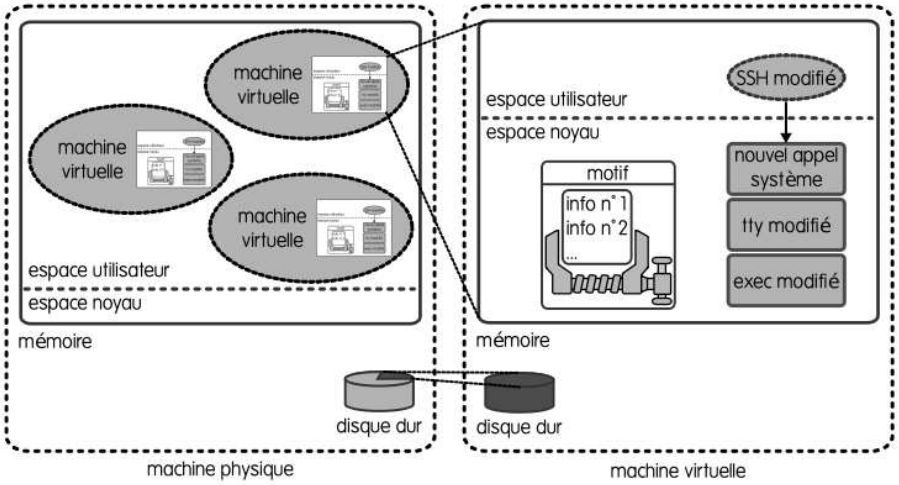

FIGURE 3.4 – (Figure 1 de [13]) Implémentation du pot de miel

Sécurité Un attaquant qui a pénétré le pot de miel ne doit pas avoir d'accès à l'extérieur, auquel cas il pourrait attaquer d'autres machines en utilisant le pot de miel comme relai (on parle alors de rebond), ce qui est illégal. Cependant, il faut maintenir un minimum de transparence pour que l'attaquant ne se doute de rien. À cette fin, deux machines virtuelles supplémentaires, accessibles uniquement depuis les trois premières, ont été mises en place pour simuler ce rebond.

3.2.3 Caractérisation d'attaques

Une fois les données collectées par un pot de miel, il faut encore les analyser. Une première observation va ainsi nous donner des indications sur les commandes les plus utilisées ainsi que les identifiants ou mots de passe les plus fréquemment tentés [14] par exemple. De plus, l'analyse de ces données va nous permettre de mieux comprendre le comportement d'attaquants qui pénètrent le système et d'améliorer ainsi les connaissances sur les risques provoqués par certains types d'intrusions. Cette partie va ainsi détailler un certain nombre de techniques utilisées pour traiter les données obtenues avec un pot de miel haute interaction et les illustrer avec des exemples de résultats obtenus dans [13].

Principales étapes d'une attaque

Pour un pot de miel, les connexions SSH issues d'une même adresse IP et rapprochées dans le temps sont regroupées en sessions afin de reconstituer l'activité d'un attaquant. Trois catégories de sessions sont ainsi distinguées :

- L'attaquant a réussi à se connecter et des commandes ont été exécutées sur le pot de miel. Ce sont des *intrusions*.
- Il n'y a pas eu de commandes exécutées, mais une grande quantité de couples (identifiant, mot de passe) ont été tentés. Ce sont des *attaques par dictionnaire*.
- Les sessions ne rentrant pas dans les deux cas précédents sont regroupées dans une dernière catégorie. Elles correspondent probablement à des « erreurs de connexions »où un utilisateur se serait rapidement rendu compte qu'il essayait de se connecter sur une autre machine que la sienne.

Il a été observé que l'intersection de l'ensemble des adresses IP réalisant des intrusions avec celui des adresses réalisant des attaques par dictionnaire est vide. Par conséquent, il existe des machines dédiées à l'exécution d'outils effectuant des attaques par dictionnaire. De plus, une intrusion sur ce pot de miel avec un couple (identifiant, mot de passe) valide est toujours précédée d'une

attaque par dictionnaire (effectuée depuis une adresse différente, donc) ayant trouvé ce couple. Il y a donc un échange d'information qui est fait par les attaquants entre ces deux phases.

Caractérisation de dictionnaires

Il est fréquent qu'un attaquant fasse appel à un outil pour tester un grand nombre de combinaisons (nom d'utilisateur, mot de passe) afin d'en trouver des valides. On parle alors d'attaque par dictionnaire et l'analyse des combinaisons tentées peut là aussi nous mener à mieux connaître les communautés d'attaquants. En effet, il est probable que des listes de ces couples fréquemment utilisés circulent parmi les attaquants (certains outils étant disponibles librement sur Internet [2]). D'après les méthodes employées dans [5], le fait de travailler sur la distance inter-textuelle, c'est à dire la similarité des vocabulaires, entre deux dictionnaires (où un dictionnaire correspond à l'ensemble des combinaisons testées lors d'une session d'attaque) nous permet de regrouper ces dictionnaires entre eux et donc d'apparenter un attaquant à telle ou telle communauté selon le dictionnaire qu'il a utilisé.
D'après les résultats de [13], il s'avère qu'en fait relativement peu de dictionnaires différents sont partagés parmi la communauté des pirates avec un grand nombre de couples utilisés qui revient souvent, ce qui signifie qu'aujourd'hui encore, beaucoup d'utilisateurs ne protègent pas leur machine avec un mot de passe complexe.

À titre d'exemple, le tableau 3.1 montre ainsi quels ont été les couples les plus tentés durant la période d'observation (419 jours).

Classification	Nombre de connexions	Identifiant	Mot de passe
1	909	test	test
2	879	admin	admin
3	864	root	root
4	824	guest	guest
5	819	root	123456
6	790	user	user
7	757	root	password
8	706	mysql	mysql
9	676	richard	richard
10	663	oracle	oracle

TABLE 3.1 – (d'après le tableau 5 de [13]) Fréquence des combinaisons les plus testées

Identification d'attaquants

Dans le cas de [13], l'objectif premier du déploiement d'un pot de miel haute interaction était l'observation d'attaquants humains. Ainsi, un certain nombre de critères peuvent être utilisés pour déterminer si les données analysées correspondent à l'action d'un homme ou d'une machine. Tout d'abord, en observant les commandes tapées au clavier, la présence de fautes de frappes est significative d'un humain (dans l'hypothèse où un script ne ferait pas exprès de se tromper). Celle-ci peut être trahie par l'entrée de commandes qui n'existent pas ou par la présence du caractère « retour arrière »signifiant qu'une correction a été apportée. De plus, si lors d'une intrusion toutes les commandes sont envoyées par blocs (et non caractère après caractère), cela sera considéré comme l'acte d'un programme.

2. Par exemple le programme John The Ripper : http ://www.openwall.com/john/

Il existe bien d'autres moyens de détecter la présence d'un humain, la piste empruntée par les pots de miels adaptatifs vus en partie 3.1.3 en est un bon exemple, puisqu'ils tentent de faire dévier l'attaquant de sa stratégie initiale, chose qu'un programme aura plus de difficultés à faire.

Activités des attaquants

L'analyse des actions menées lors des intrusions a permis de relever certains comportements typiques lors d'une attaque. Tout d'abord, les attaquants changent le mot de passe du compte qu'ils ont piraté afin de pouvoir être les seuls à l'utiliser. Leur objectif suivant est de télécharger des logiciels malveillants. Pour ce faire, ils tentent très souvent d'utiliser la commande `wget`. Les connexions sortantes du pot de miel étant bloquées, un grand nombre d'attaquants (visiblement inexpérimentés) arrêtent ici leur activité alors qu'il est possible d'utiliser d'autres commandes pour contourner ce problème. La décompression des logiciels téléchargés se fait ensuite dans un répertoire « caché »généralement créé dans un répertoire standard (type `/var/tmp`) du système afin de dissimuler leur présence. Ces programmes sont généralement de trois types :

- Des applications effectuant un scan de ports afin de détecter d'autres machines accessibles. La machine attaquée est alors utilisé comme rebond par le pirate pour attaquer des nouvelles cibles.
- Des clients `irc` pour pouvoir ensuite transmettre à la machine infectée des instructions à distance. La machine se voit ainsi rattachée à un *botnet*, c'est à dire un groupe de machines compromises qui vont être utilisées afin de réaliser massivement des tâches malveillantes à l'insu de leur propriétaire (attaques par déni de service ou envoi de pourriels par exemple).
- Des programmes exploitant des vulnérabilités du système afin d'obtenir les privilèges du super-utilisateur. Étonnamment, peu d'attaquants ont essayé de faire ce genre de manipulation.

3.3 Déroulement des travaux

3.3.1 Objectifs

Les expériences précédemment effectuées au LAAS n'utilisaient qu'un pot de miel de ce type, installé au laboratoire. Durant ce stage, notre objectif sera tout d'abord de concevoir un système permettant le déploiement de tels pots de miel sur des réseaux se trouvant en d'autres endroits, ainsi que le rappatriement des données collectées par ceux-ci. Par suite, nous devrons proposer et effectuer des analyses sur les nouvelles données afin de confirmer ou infirmer les tendances décrites dans [5], voir d'en faire émerger de nouvelles. De plus nous confronterons entre elles les informations fournies par ces nouveaux pots de miel afin de mettre en évidence des similarités ou disparités pouvant apparaitre lors des attaques en fonction de leur cible.

Pour atteindre nos objectifs, il est nécessaire de réaliser diverses tâches lors de ce stage. Ainsi, nous devons tout d'abord concevoir l'architecture du réseau qui nous permettra de déployer les pots de miel en différents endroits. Il faut également créer une base de données pour accueillir les informations fournies par ces pots de miel, puis définir et concevoir un système permettant d'accéder simplement à ces données pour leur traitement. Grâce à ces outils, nous pourrons enfin analyser les données collectées pour tenter de répondre à notre problématique.

3.3.2 Déroulement de l'expérience

Les informations que nous cherchons à obtenir concernent le comportement des attaquants ayant réussi à pénetrer dans un système via le protocole `ssh`. Or, les impératifs du stage font que la durée des expériences va être relativement courte comparée à la précédente [13], nous devons nous assurer que plusieurs attaquants vont réussir à pénetrer notre système rapidement

afin d'avoir le temps de mener des analyses. Pour ce faire, nous avons décidé de séparer notre expérimentation en deux phases.

- Lors d'une première phase d'une durée d'un mois, nous allons ouvrir nos pots de miel sans aucun compte accessible. Nous allons seulement collecter des informations récentes sur les couples identifiant/mot de passe les plus tentés par ceux qui ciblent nos pots de miel. En fonction de ces données, nous allons constituer une liste de comptes ayant une forte probabilité de se faire exploiter lors de futures attaques. L'intérêt de ce processus est que nous pourrions observer des cibles privilégiées par les attaquants auxquelles nous n'aurions pas pu penser. En leur offrant ces cibles, nous pourrons peut être comprendre leurs objectifs justifiant leur attrait pour celles-ci.
- La deuxième phase commence dès la création des comptes. Nous pouvons donc récupérer des informations sur les intrusions qui se produiront, tout en continuant de collecter les données relatives aux tentatives de connexions `ssh`.

Chapitre 4

Architecture de collecte des données

Notre objectif vise le déploiement des pots de miel à divers endroits dans le monde afin de pouvoir comparer les comportements des différents attaquants observés. Nous voulons déterminer si des tendances globales se dessinent ou si les activités sont, au contraire, très dépendantes de la situation géographique du pot de miel. Pour cela, nous avons pu avoir accès à trois machines, chacune possédant une adresse IP publique, installées en trois sites différents : une à Toulouse, une à Rennes et une à College Park, Maryland (États-Unis). Afin d'avoir un site dont nous contrôlons tous les paramètres, nous avons également décidé d'installer un ordinateur au LAAS, qui servira aussi de pot de miel et devra être configuré d'une manière identique aux trois autres. Ce chapitre présente l'architecture qui a été conçue lors de ce stage pour permettre à ces pots de miel de fonctionner.

4.1 Architecture déployée

4.1.1 Présentation générale

Le déploiement de pots de miel haute interaction peut être réalisé selon deux approches. Une première approche pourrait être de déporter l'architecture des pots de miel utilisés lors des expériences précédentes, c'est à dire d'installer nos machines virtuelles modifiées sur un ordinateur en ces différents sites. Cette approche souffre de trois problèmes majeurs. Tout d'abord, comme expliqué en partie 3, le déploiement d'un pot de miel haute interaction reste une opération risquée (puisqu'il est destiné à laisser réellement un attaquant opérer sur le système). Il est donc soumis à de nombreuses contraintes de sécurité. Par conséquent, multiplier les systèmes revient à multiplier ces contraintes, d'autant plus que nous ne connaissons pas exactement la configuration des réseaux sur lesquels nous souhaitons installer les pots de miel. Ensuite, tous les pots de miel doivent être déployés dans des conditions identiques pour éviter que des différences de paramétrage nuisent à l'expérimentation. Notre architecture doit donc être pensée en fonction de ce besoin. Enfin, nous ne souhaitons pas que les réseaux des partenaires sur lesquels vont se connecter les attaquants puissent être compromis. Il est donc exclu de permettre l'exécution des instructions de l'attaquant sur des machines qui ne sont pas les nôtres. Notre système devra donc faire croire à un attaquant qu'il se connecte sur un ordinateur situé à Toulouse, Rennes ou College Park alors qu'il interagira en fait avec une machine du LAAS.

Ainsi, nous avons adopté une approche pour laquelle les connexions d'un attaquant vont être redirigées vers des machines virtuelles installées sur une machine du LAAS. Cette approche ne souffre pas des trois problèmes précédents. De plus, elle nous permet de garder un contrôle suffisamment important sur les pots de miel pour garantir un niveau de sécurité satisfaisant ainsi qu'une meilleure capacité de réaction en cas de problème (les éléments « sensibles »de l'infrastructure étant facile d'accès). De plus, cela nous permet de minimiser l'influence du matériel et des

réseaux mis à notre disposition dans les divers lieux de déploiement en n'utilisant les machines distantes uniquement comme des relais vers notre installation locale.

La figure 4.1 présente un aperçu global de l'architecture de ce système. Nous avons ainsi trois machines situées sur des réseaux distants, plus une installée au LAAS. Chacune possède une IP publique. Elles vont servir de relai vers nos machines virtuelles, notées VM1 à VM4. Ces dernières sont installées sur un unique hôte. Cet hôte simule également un environnement de réseau local distinct pour chacune des machines virtuelles. Des tunnels sont créés entre les relais et l'hôte, qui se charge de faire le lien entre ses extrémités de tunnels et les passerelles virtuelles des machines virtuelles. Nous devrons toutefois faire en sorte que les réponses envoyées par les machines virtuelles aux attaquants suivent bien le même chemin que les requêtes lancées par ces attaquants (à savoir passer par les tunnels pour ressortir par le relai), sinon elles risquent d'être rejetées par des pare-feux d'Internet. Enfin, l'établissement de règles de routage au niveau des relais et de l'hôte va permettre à une partie du trafic ciblant les IP publiques des relais d'être redirigée par ceux-ci vers les machines virtuelles. Nous allons maintenant expliquer plus en détail le fonctionnement de cette architecture.

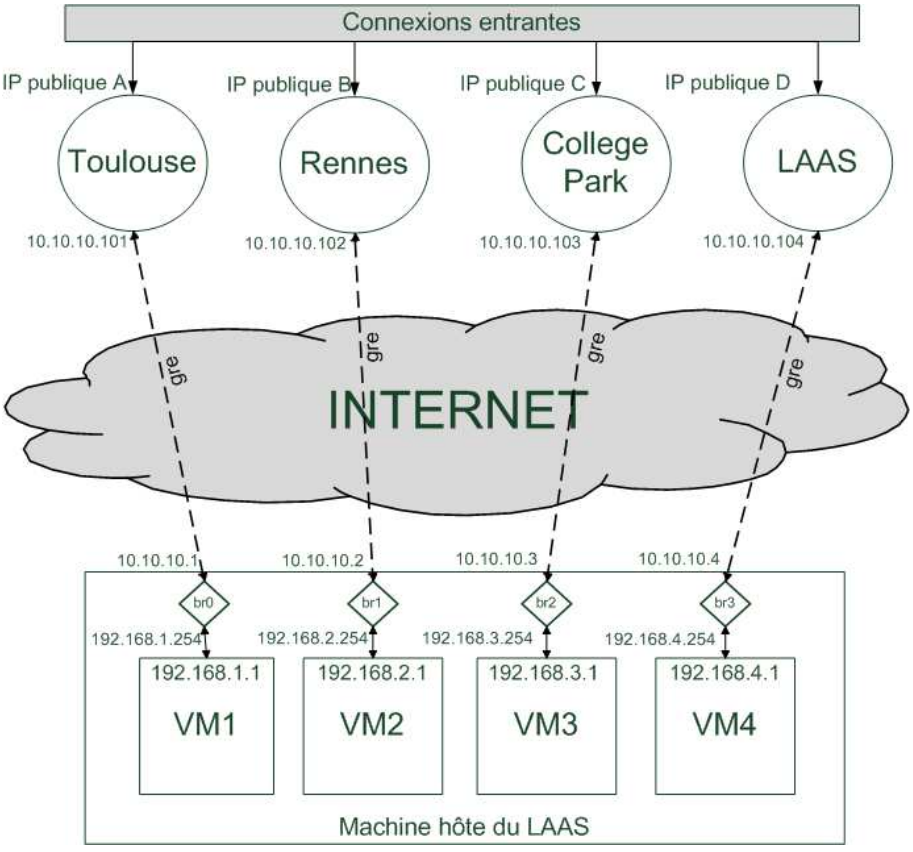

FIGURE 4.1 – Schéma de l'architecture du système de pots de miel

4.1.2 *Tunneling*

L'ordinateur hébergeant les machines virtuelles ne possède qu'une seule IP publique. Par conséquent, nous avons attribué des adresses IP privées aux machines virtuelles. Pour assurer une communication directe entre celles-ci et leur relai, nous ne pouvons donc pas directement « rerouter »les paquets arrivant sur un relai vers l'IP de la machine virtuelle correspondante, celle-ci n'étant pas publique. Le problème revient donc à établir une connexion directe entre deux machines de deux réseaux différents connectés à internet. Notre solution est d'établir des tunnels

(en pointillés sur la figure 4.1) entre la machine physique du LAAS et chacun des relais. Ainsi l'hôte pourra faire le lien entre un relai et sa machine virtuelle sans se soucier des adresses non routables.

Pour établir ces tunnels, nous utilisons le protocole `GRE` (*Generic Routing Encapsulation*), décrit dans la RFC 2784[1], qui permet d'encapsuler des protocoles dans IP. Il va nous permettre de créer un réseau privé virtuel (`VPN`, *Virtual Private Network*) entre deux machines, situées sur deux réseaux différents, dont une possède une adresse privée. La figure 4.2 illustre l'encapsulation pour notre architecture : les éléments grisés correspondent aux données encapsulées dans `GRE`, ce qui nous donne une pile protocolaire portée par une autre (les éléments blancs) à travers Internet.

FIGURE 4.2 – Encapsulation d'une trame SSH passant dans nos tunnels GRE

Au LAAS, quatre machines virtuelles s'exécutent sur une unique machine physique disposant d'une unique adresse publique. C'est par cette adresse que transiteront les données provenant de tous les relais avant d'arriver aux machines virtuelles. La machine hôte devra donc se comporter comme une passerelle pour faire le lien entre chaque machine virtuelle et les extrémités des tunnels. Chaque machine virtuelle est dotée d'une adresse IP privée (192.168.0.0/16). Cela rend plus crédible l'environnement attaqué et nous permettra également d'ajouter facilement des machines virtuelles au réseau puisque l'infrastructure sera déjà prête. Pour ce faire, nous avons créé des ponts virtuels Ethernet (appelés `br` sur la figure 4.1), qui serviront de passerelle pour le réseau local auquel ils sont rattachés. Une machine virtuelle est alors rattachée à chacun de ces ponts.

4.1.3 Contrôle du trafic

Contrôle des connexions

Le contrôle des connexions intervient à deux endroits : au niveau des machines relais et au niveau de la machine hôte du LAAS. Nous effectuons ce contrôle au moyen de l'outil `iptables` qui nous permet d'établir des règles de filtrage sur les paquets entrants et sortants d'une machine administrée par un système d'exploitation Linux. Ce filtrage des paquets est géré dans le noyau Linux par le module `netfilter`. Notamment, ce module présente l'avantage de faire du suivi d'état des connexions TCP et permet donc une analyse contextuelle et non isolée de chaque segment TCP. Autrement dit, le filtrage s'établit non seulement sur les adresses IP et numéros de port, mais aussi sur les flags TCP, numéros de séquence et acquittement. Cette gestion simplifie de plus l'écriture des règles de filtrage.
La politique de sécurité associée aux machines relais est la suivante :

- Les connexions entrantes sur le port 22 sont autorisées. C'est le port par défaut pour `ssh`, et donc celui sur lequel les attaquants vont pouvoir se connecter.
- Les connexions sortantes sur les ports 53 (`dns`) et 123 (`ntp`) sont autorisées pour que nos machines virtuelles puissent utiliser ces services sans éveiller de soupçons de la part de l'attaquant pour autant. Les machines virtuelles peuvent ainsi utiliser les serveurs ntp et dns des réseaux de leurs relais respectifs, comme le ferait un véritable ordinateur présent sur ce réseau.
- Un port supplémentaire (2222) est ouvert uniquement pour les machines possédant une IP du LAAS. Un serveur `ssh` écoute sur ce port et permet ainsi d'administrer la machine à distance.

1. http ://www.ietf.org/rfc/rfc2784.txt

- Le protocole `GRE` est autorisé en entrée et en sortie.
- Le protocole `icmp` est autorisé en entrée et en sortie en direction de l'autre extrémité du tunnel. En effet, du suivi d'état peut être effectué sur des réseaux, coupant automatiquement une connexion vers une de leurs machines si il n'y a pas eu d'activité sur cette dernière depuis un certain temps. Or, ces réseaux n'étant pas sous notre contrôle, nous ne pouvons pas modifier leurs paramètres. Pour éviter ceci, et nous assurer que les tunnels n'ont pas été coupés, nous envoyons un ping d'une extrémité vers l'autre toutes les minutes.
- Toutes les autres tentatives de connexions (entrantes et sortantes) sont rejetées.

La politique de sécurité au niveau de l'hôte des machines virtuellesest la suivante :

- Le protocole `GRE` est autorisé en entrée et en sortie.
- Les connexions `ssh` sur le port 2222 sont autorisées depuis le LAAS, pour administrer la machine.
- Les connexions `ssh` sur le port 22 sont autorisées des tunnels vers les machines virtuelles.
- `dns` et `ntp` sont autorisés depuis les machines virtuelles vers les tunnels.
- `icmp` est autorisé vers les tunnels.

Routage des connexions

Comme décrit dans la section 4.1.2, un réseau privé virtuel est créé entre une machine relai et une machine virtuelle en utilisant un tunnel `GRE`. Ainsi, pour qu'une attaque tentée contre un relai puisse s'effectuer en réalité sur le pot de miel, il faut que le relai redirige automatiquement les requêtes arrivant sur son port 22 vers le port 22 de la machine virtuelle se trouvant à l'autre bout du tunnel. Nous utilisons pour cela la table `nat` (*Network Adress Translation*, traduction d'adresse réseau) de `iptables` afin de changer l'IP (publique) de destination des paquets entrants pour celle (privée) de la machine virtuelle. De même, toute nouvelle requête sortante autorisée (`dns`, `ntp` ou les réponses aux connexions `ssh` déjà établies) verra son adresse d'origine, donc celle de la machine virtuelle, traduite vers l'adresse publique du relai. Ensuite, il faut mettre à jour les tables de routage du relai et de l'hôte des VM pour que les paquets puissent transiter par le tunnel. Par exemple, pour qu'une connexion lancée sur le relai 1 soit établie avec la VM 1, nous devons créer les règles de routage suivantes :

- Au niveau de la table de routage du relai, si 10.10.10.1 est l'adresse de la machine hôte à travers le tunnel `GRE` et 192.168.1.1 est l'adresse de la VM 1 dans son réseau local, alors on a :

`Destination`	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
`10.10.10.1`	*	255.255.255.255	UH	0	0	0	tun
`192.168.1.0`	10.10.10.1	255.255.255.0	UG	0	0	0	tun

- Au niveau de la table de routage de l'hôte, si 192.168.1.0 est le réseau privé de la VM 1 et 10.10.10.101est l'adresse du relai de la machine 1 à travers le tunnel `GRE`, alors on a :

`Destination`	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
`10.10.10.101`	*	255.255.255.255	UH	0	0	0	tun0
`192.168.1.0`	*	255.255.255.0	U	0	0	0	br0

Ainsi, une connexion arrivant sur le relai de la VM 1 (donc à destination de l'adresse publique du relai) verra sa destination changée par l'adresse de la VM 1 (192.168.1.1) et sera alors redirigée vers l'interface tun, qui représente une extrémité du tunnel. Elle arrivera donc à la machine hôte par l'interface tun0 (soit l'autre bout du tunnel). Sa cible étant la VM 1, elle passera alors par l'interface br0 (192.168.1.254), soit la passerelle du réseau local de VM 1, et sera donc réceptionnée par la machine.

4.2 Collecte et traitement des données

Dans cette partie nous allons décrire les méthodes utilisées pour récupérer et trier les données issues des pots de miel en vue de leur analyse future.

4.2.1 Nature des données

Comme détaillé en partie 3.2, nos pots de miel possèdent un noyau dont certains appels systèmes ont été modifiés afin d'enregistrer les données qu'ils interceptent. Ainsi, les machines virtuelles stockent dans leur mémoire les informations concernant :

- Les tentatives de connexions `ssh` effectuées lors d'une attaque. Nous pouvons ainsi connaître l'adresse IP de l'attaquant et les couples (identifiant, mot de passe) qui ont été tentés.
- Le contenu des terminaux ouverts par l'attaquant. Cette information nous permet de voir ce que l'attaquant écrit, comment il l'écrit (en tapant lui même les commandes ou en copier/coller) et ce que ses terminaux lui affichent.
- La liste des fichiers exécutés lors de l'attaque. Nous savons ainsi quels programmes sont exécutés lors d'une intrusion, avec leurs paramètres.

Ces données sont extraites de la mémoire des machines virtuelles une fois par jour et conservées sur la machine hôte sous forme de fichiers binaires classés selon leur date d'extraction. La mémoire des machines virtuelles dédiée à ce stockage est ensuite vidée. Or, malgré l'existence de programmes développés pour lire et afficher directement le contenu de ces binaires, le format de ces derniers n'est pas pratique dès lors que l'on cherche à faire des analyses plus poussées. En effet, il faudrait alors convertir à chaque fois les fichiers pour pouvoir traiter leur contenu ou encore parcourir un grand nombre de dossiers (un par jour) lorsque l'on cherche à comparer des données entre elles. De plus, il n'est pas aisé d'avoir une vision globale sur l'ensemble des données à moins de passer par de nombreux scripts et fichiers pour formater ces données brutes. Par conséquent, nous avons décidé d'utiliser une base de données pour stocker ces informations dans un format permettant de faciliter l'accès aux données et les analyses qui vont suivre.

4.2.2 Structure de la base

Quatre bases de données, une pour chaque machine, ont été créées. Ces bases sont toutes conçues de la même manière. Les données sont stockées sur un serveur `MySQL` installé sur un ordinateur dédié au stockage et à l'analyse des données.

Lors de leur conception, nous avons eu affaire à une contrainte importante. En effet, les informations relatives à une attaque étant enregistrées par différents programmes, le format des données brutes n'est pas uniforme. Nous avons alors cherché une structure nous permettant de rapprocher les données relatives à une même attaque sans perdre d'information stockée dans les binaires. Nous sommes donc arrivés à la structure de données représentée en figure 4.3.

La table nommée `op_hdr` correspond aux informations brutes collectées depuis la mémoire des pots de miel et contient les informations génériques de chaque donnée. Cette table contient une ligne pour tous les éléments de la base. En effet, chaque entrée dans la base a une `op_id` unique qui est donc répétée ici. On peut ainsi rattacher toute entrée d'une autre table à une de `op_hdr` via leur `op_id`. La signification des différents champs est la suivante :

- `op_id`, unique pour chaque entrée, est donc la clé primaire de cette table.
- `index_op` correspond à l'indice décrit dans la section 3.2.2. Il s'incrémente à chaque nouvel évènement et permet de détecter les pertes d'information ainsi que d'établir une chronologie plus précise que celle fournie par la colonne `time`, qui n'a qu'une précision de l'ordre de la seconde.
- `time` est la date (à l'heure Unix) de l'évènement.
- `op_type` indique la table qui contient les détails de l'entrée avec l'`op_id` correspondante.

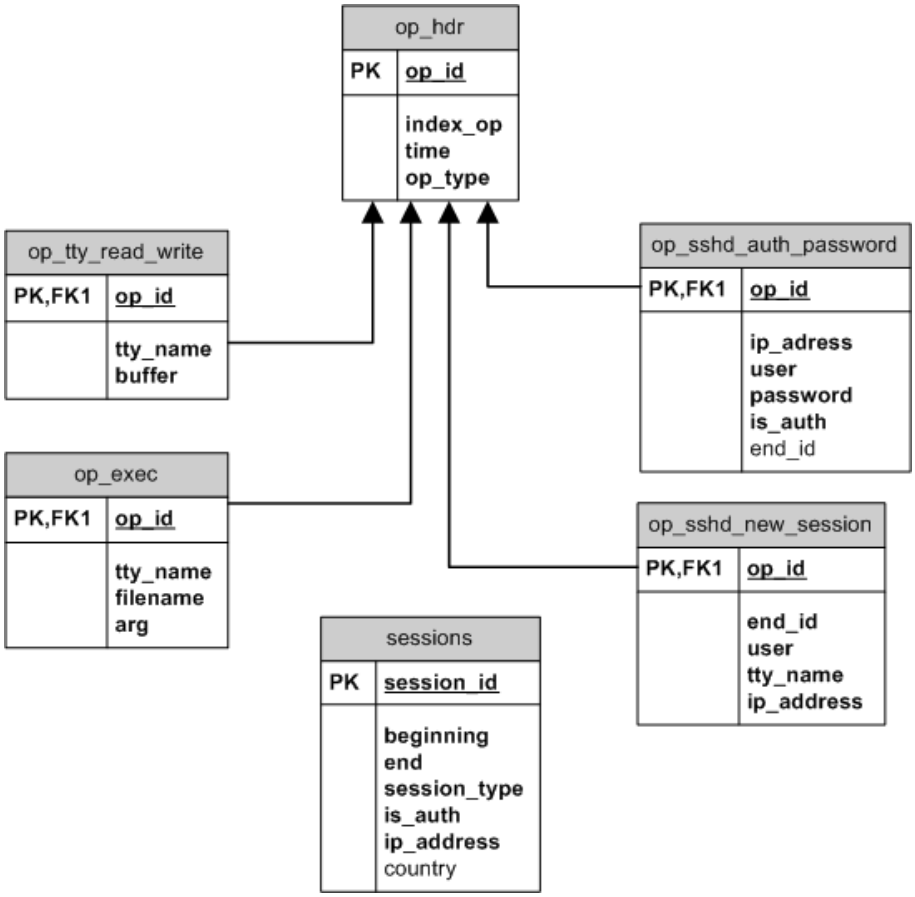

FIGURE 4.3 – Schéma de la base de données d'un pot de miel

La table `op_sshd_auth_password` contient les données relatives à toutes les tentatives de connexion ssh. La signification des différents champs est la suivante :

- `ip_address` est l'adresse IP source de l'attaque.
- `user` est l'identifiant utilisé lors d'une tentative.
- `password` est le mot de passe tenté avec l'identifiant
- `is_auth` indique si la tentative de connexion a réussi (identifiant et mot de passe corrects).
- `end_id` indique, si la connexion a réussi, l'identifiant de la dernière action répertoriée lors de cette attaque.

La table `op_sshd_new_session` est renseignée quand une connexion `ssh` a abouti. Relativement similaire à `op_sshd_auth_password`, elle ne contient que des informations sur les attaques pendant lesquelles l'ouverture d'un terminal a été requise. Une colonne importante est `tty_name`, qui indique le terminal attribué à l'attaquant. Cette information va nous servir à relier les données fournies par le pilote `tty` modifié à celles collectées par le serveur `ssh`.

La table `op_tty_read_write` contient toutes les informations enregistrées par le pilote `tty`. Une ligne contient ainsi le contenu du buffer de `tty` ainsi que le terminal auquel il est destiné. Lorsque les données concernent des entrées faites au clavier par l'attaquant, le buffer ne contiendra qu'un caractère à la fois. En revanche, si le texte a été copié/collé, le buffer contiendra plusieurs caractères. Il est ainsi facile de faire une première observation du comportement de l'attaquant.

La table `op_exec` contient les données concernant les programmes exécutés par la machine lors d'une intrusion. Une entrée correspond au nom du programme exécuté, aux arguments qui lui sont passés et au terminal dans lequel il s'est exécuté.

Une fois la base remplie, des tables `sessions` peuvent être créés et mises à jour. Ces tables

contiennent les données issues du regroupement des connexions `ssh` en sessions d'attaques (telles que définies en partie 3.2). Ces tables vont ainsi contenir :

- `beginning` et `end` indiquent les identifiants des évènements de début et de fin d'une session.
- `session_type` indique si l'attaque est une intrusion, une attaque par dictionnaireou aucune des deux.
- `is_auth` vaut 1 si au moins une tentative de connexion a réussi lors de la session, 0 sinon.
- `ip_address` correspond à l'adresse IP source de l'attaque.
- `country` indique le pays correspondant à cette IP.

La méthode utilisée pour compléter ces tables est décrite dans la section suivante.

Par ailleurs, nous avons également importé grâce à ces mêmes scripts les données recueillies par le pot de miel précédemment déployé au LAAS, sur une période allant de janvier 2006 à août 2010. Celles-ci nous seront utiles pour effectuer des comparaisons avec les nouvelles expériences.

4.2.3 Interface de gestion

Renseignement de la base

De nouveaux fichiers de données (un par machine virtuelle) sont générés par les pots de miel une fois par jour à 6h, selon la méthode décrite en 3.2.2. Utiliser ces binaires pour mettre la base de données à jour nécessite d'effectuer plusieurs manipulations. Certaines sont nécessaires pour détecter et corriger les problèmes qui peuvent survenir lors de l'enregistrement des données. D'autres servent à mettre à jour les tables de la base. Nous avons ainsi écrit plusieurs scripts en `Perl` (avec le module `DBI`, permettant de dialoguer avec une base de données) afin de lire et modifier le contenu des bases de données. Ce langage a été choisi car il permet une manipulation simple et rapide de chaînes de caractères, propriété qui nous sera utile pour analyser le contenu des sessions enregistrées.

Par conséquent, une fois par jour, après la génération de nouveaux fichiers par le pot de miel, un script `bash` que nous avons écrit exécute ces scripts pour mettre à jour la base. Il fonctionne ainsi :

- Copie des nouveaux binaires sur l'ordinateur contenant la base de données.
- Génération des fichiers `sql` correspondants et mise à jour de la base : nous avons modifié un programme permettant d'afficher le contenu de ces fichiers pour qu'il génère un fichier au format `.sql` avec les données formatées selon la structure de la base. Ce fichier est ensuite lu par `MySQL` pour compléter la base de données.
- Suppression des doublons ayant pu apparaître : au niveau des binaires fournis par le pot de miel, il peut arriver que le compteur indiçant les données soit remis à zéro ou que des données soient dupliquées en mémoire. Cela arrive notamment lors d'un redémarrage de la machine virtuelle. Nous avons écrit un script qui parcourt les nouvelles données ajoutées à la table `op_hdr` et regarde si l'indice `index_op` reste croissant. Si ce n'est pas le cas, il supprime les données nouvellement ajoutées correspondantes si le reste de la ligne est identique aux données plus anciennes de même indice.
- Mise à jour de la table des sessions d'attaque : ces sessions sont calculées en deux étapes. Les tentatives de connexions `ssh` sont regroupées grâce à un algorithme de fenêtre glissante. Plus précisément, nous fixons une valeur seuil et si deux évènements consécutifs provenant de la même IP sont séparés par une durée inférieure à cette valeur, alors on considère qu'ils font partie de la même session. Si une des tentatives ainsi rapprochées a réussi et qu'une intrusion a eu lieu, la date du dernier événement de la session devient alors la date de fin de l'intrusion, et le prochain regroupement sera calculé depuis cette date.

Un exemple de données à traiter est fourni figure 4.4. Chaque point correspond à une tentative de connexion. L'algorithme va créer trois sessions entre t0 et t1. En effet, au vu de la durée du seuil donnée, l'écart entre le dernier point de la session A et le premier de la session C est

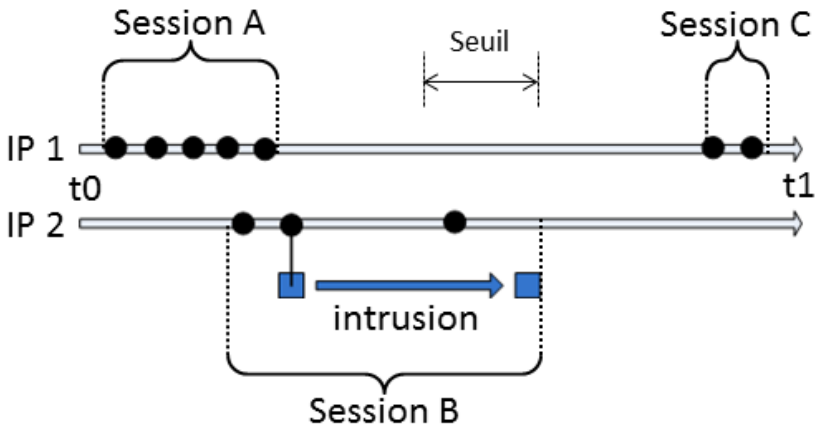

FIGURE 4.4 – Exemple de séparation en sessions par l'algorithme de fenêtre glissante

bien supérieur à ce seuil. La séparation est donc cohérente. Sur la seconde ligne, les tentatives de connexion provenant d'une autre IP ont lieu en même temps que celles de la session A. Cependant, leur source étant différente, ces tentatives ne sont pas rattachées à ces dernières mais initient une nouvelle session. En revanche, la deuxième tentative de connexion de l'IP2 a réussi et une intrusion, dont la durée est représentée par l'espace entre les carrés, s'est produite. Dès lors, toutes les tentatives de connexion de cette IP ayant lieu pendant l'intrusion sont rattachées à cette session, même si l'écart entre celles-ci est supérieur au seuil.

Visualisation

Afin de visualiser plus clairement les données d'attaque dans leur ensemble, nous avons développé une interface graphique grâce au module `Tk` pour `Perl`. Notre objectif était d'avoir une interface relativement simple d'utilisation qui, en partant d'une vision globale des sessions, nous permettrait d'observer plus en détails des éléments de notre choix. Celle-ci peut se décomposer en deux parties principales.

- Tout d'abord, la moitié supérieure de l'interface permet de sélectionner ce que l'on souhaite afficher grâce à un ensemble de boutons et de champs de saisie. Ainsi, dans la figure 4.5, nous avons par exemple sélectionné en quelques clics l'ensemble des sessions relevées par les pots de miel de Rennes et Toulouse pendant le mois de juin, effectuées par des IP vues sur le pot de miel du LAAS mais pas sur celui de College Park. Les cases renseignées permettent alors de générer automatiquement une ou plusieurs requêtes `SQL` en fonction du nombre de bases à interroger.
- Le tableau présent dans la moitié inférieure permet d'afficher les résultats correspondant à notre requête dans un tableau, puis d'interagir avec ceux-ci. Il nous donne de plus des informations d'ordre général sur les résultats de notre recherche, telles que le nombre d'adresses différentes répertoriées ou le nombre de sessions que l'on a sélectionnées. Il est possible de trier ce tableau selon n'importe lequel de ses attributs, par ordre croissant ou décroissant. Par défaut, les sessions sont classées par ordre chronologique. La figure 4.5 donne un exemple d'affichage des résultats obtenu avec les paramètres donnés en exemple précédemment.
- Enfin, la partie basse de l'interface comporte plusieurs boutons permettant d'afficher ou d'exporter le contenu des sessions sélectionnées. Il est possible d'obtenir l'ensemble des couples tentés lors de l'étape de connexion ssh, le contenu des terminaux vus par les atta-

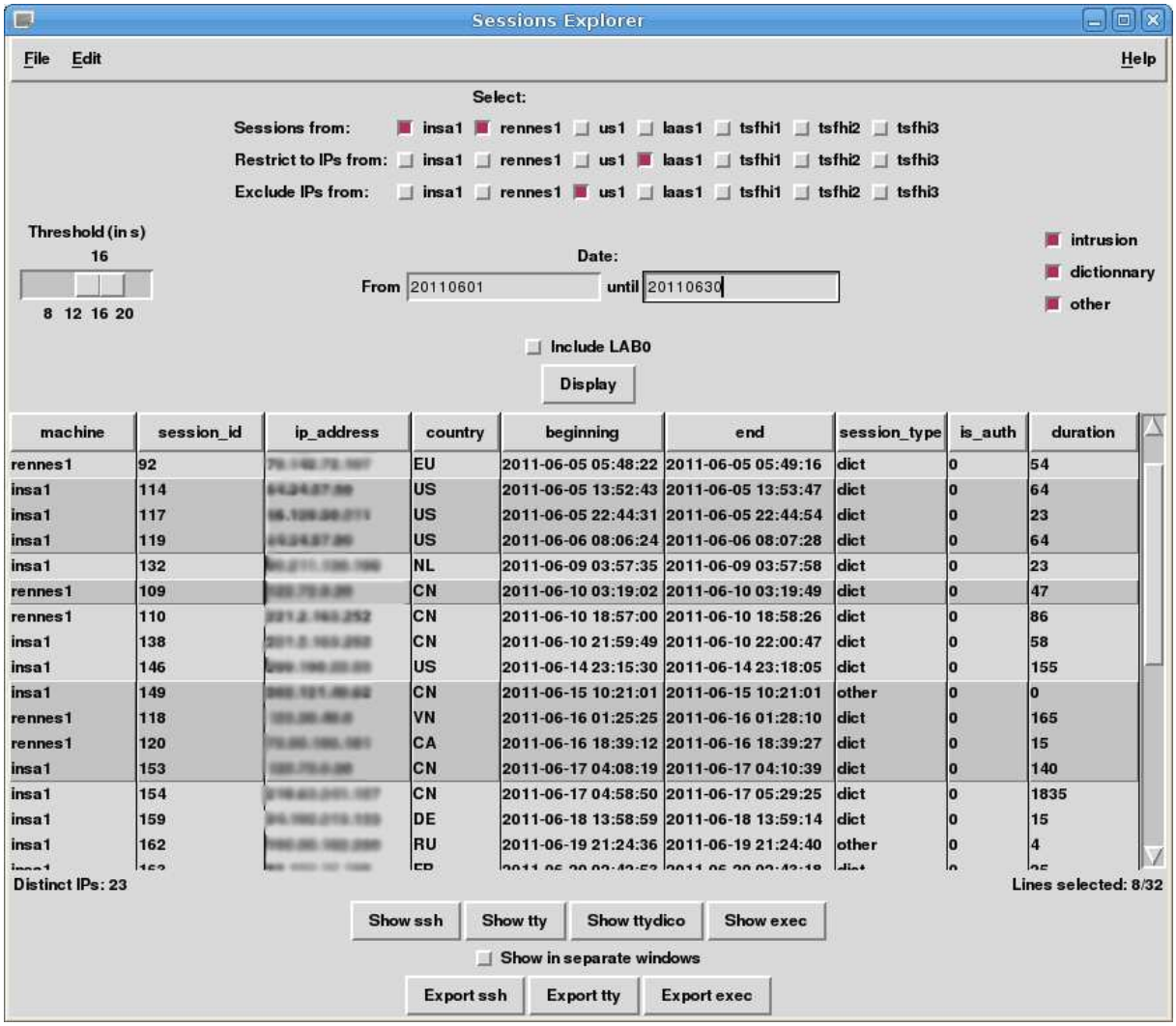

FIGURE 4.5 – Aperçu général de l'interface de visualisation des données

quants ainsi que les commandes qu'ils ont tapées ou encore la liste des programmes exécutés lors de la session. De plus, un dernier bouton « `ttydico` »permet d'obtenir un vocabulaire (dans un format semblable à celui de la liste des couples tentés) des commandes entrées par l'attaquant lors de l'intrusion. Cette dernière fonctionnalité n'est pas encore implémentée et sera discutée plus en détail dans la partie 5.4.

Dans son état actuel, cette interface nous permet d'obtenir rapidement des aperçus d'ensemble sur tout ou partie de nos données, ce qui en fait un outil précieux pour l'exploration et la compréhension de celles-ci. Elle a notamment été employée pour mener les analyses présentées dans le chapitre suivant.

Chapitre 5

Analyse des données

Notre expérimentation s'est déroulée en deux phases. Tout d'abord, nous avons déployé nos pots de miels sans créer de comptes sur ceux-ci. Ils ont alors été laissés dans cet état pendant un mois. Cette première phase s'est déroulée du 1er au 30 juin. L'analyse des données collectées durant cette phase a permis d'avoir un aperçu des couples identifiant/mot de passe tentés par les attaquants ainsi que leur fréquence. Ces résultats ont été utilisés pour constituer une liste de comptes que nous savions régulièrement tentés. La seconde phase a débuté par la création des comptes précédents sur tous les pots de miel. Grâce à l'utilisation de ces comptes fréquemment tentés, nous espérions pouvoir observer rapidement des intrusions. Cette deuxième phase a commencé le 1er juillet à 13h00. Dans ce chapitre, nous allons d'abord donner un aperçu d'ensemble des résultats obtenus lors de ces deux phases, en les distinguant si besoin. Ensuite nous nous intéresserons à quelques analyses plus poussées sur ces données. Pour finir, nous commenterons les travaux en cours au moment de la rédaction de ce rapport.

5.1 Activités observées

Dans cette partie, nous analysons les résultats des observations sur les différents pots de miel en amont de tout traitement des données. Ces analyses concernent les deux phases de l'expérimentation.

5.1.1 Connexions `ssh`

Aperçu

Le tableau 5.1 donne la répartition des connexions observées sur les différents pots de miel à la date du 1er août. Chacune de ces connexions correspond à l'envoi d'un couple identifiant/mot de passe au serveur `ssh`.

Pot de miel	Nb. connexions	Nb. connexions réussies	Nb. IP différentes
Toulouse	160764	57	197
Rennes	114520	114	153
College Park	6510	3	63
LAAS	14496	109	181
Total	296290	283	501

TABLE 5.1 – Répartition des connexions `ssh` observées sur les pots de miel.

Le nombre d'adresses distinctes obtenu en considérant tous les pots de miel ensemble étant plus faible que la somme des nombres obtenus pour chaque machine. Certaines IP se sont donc

connectées sur plusieurs d'entre elles. Il faudra donc tenir compte de ces intersections lors de nos analyses transversales.

Les tableaux 5.2 et 5.3 donnent les couples les plus tentés lors de ces attaques sur chaque pot de miel au bout de deux mois. Sans surprise, le compte `root` (le compte administrateur sous Linux) est la cible privilégiée des attaques, et ce quel que soit le pot de miel considéré. Pour le reste, nous voyons que les couples les plus tentés sont simples, avec un mot de passe très souvent équivalent au nom d'utilisateur. Ce point confirme les observations faites il y a quatre ans dans [5]. On peut donc supposer que l'emploi de tels mots de passe pour protéger ses données est encore très répandu (un attaquant n'aurait pas d'intérêt à tester de telles combinaisons si elles étaient très rares). De plus, il semble y avoir un attrait pour les identifiants correspondant à des comptes créés pour le besoin de certains programmes (oracle, mysql, postgres, nagios, etc.). Les mots de passe associés sont ceux utilisés par défaut lors de l'installation de ces applications. Il serait intéressant de savoir si ces applications sont ciblées seulement parce qu'elles sont très répandues ou si l'intérêt est également d'accéder aux fichiers qu'elles utilisent et aux données qu'elles manipulent.

En comparant les résultats obtenus sur chaque pot de miel, nous constatons des variations du classement, avec toutefois certains couples présents dans les premières positions sur tous les pots de miel.

	Toulouse		Rennes		College Park		LAAS	
Classement	Couple	Nb.	Couple	Nb.	Couple	Nb.	Couple	Nb.
1	root 123456	182	root 123456	132	root root	21	root root	95
2	oracle oracle	152	root password	111	root 123456	19	root 123456	81
3	root password	133	test test	109	test test	18	root qwerty	76
4	test test	131	oracle oracle	102	root qwerty	15	root password	64
5	root qwerty	121	postgres postgres	99	root 12341234	15	root 111111	63

TABLE 5.2 – Liste des 5 couples les plus tentés sur chaque pot de miel

	Toulouse		Rennes		College Park		LAAS	
Classement	Couple	Nb.	Couple	Nb.	Couple	Nb.	Couple	Nb.
1	oracle oracle	152	test test	109	test test	18	oracle oracle	48
2	test test	131	oracle oracle	102	admin admin	10	test test	40
3	mysql mysql	115	postgres postgres	99	oracle oracle	10	postgres postgres	32
4	postgres postgres	100	mysql mysql	95	cary cary	9	nagios nagios	27
5	test test123	90	user user	77	carlos carlos	8	mysql mysql	23

TABLE 5.3 – Liste des 5 couples hors `root` les plus tentés sur chaque pot de miel

Paramétrage

À la fin de la première phase, nous avons regardé quels étaient les couples les plus tentés sur chaque pot de miel. En nous basant sur les éléments de cette liste (qui exclut `root`), nous avons déterminé une liste de couples identifiant/mot de passe à utiliser sur les quatre pots de miel. Les couples composant cette liste ont été choisis selon plusieurs critères. Ainsi, dans l'ensemble des couples les plus tentés, nous avons retenu :

- Des couples parmi les plus tentés sur un pot de miel mais pas sur les autres
- Des couples fréquemment tentés sur tous les pots de miel
- Des couples pour lesquels l'identifiant est différent du mot de passe
- Des couples pour lesquels l'identifiant est identique au mot de passe
- Des couples dont l'identifiant correspond à une application pouvant être installée sur la machine (`apache`, `mysql`, etc.)

À partir de ces critères, nous avons créé les comptes suivants sur tous les pots de miel :

Compte	Identifiant	Mot de passe
C1	adam	adam
C2	alex	alex123
C3	apache	apache
C4	cary	cary
C5	eric	eric
C6	michael	michael
C7	mysql	mysql
C8	nagios	123456
C9	postgres	postgres
C10	test	test123
C11	user	password

TABLE 5.4 – Ensemble des comptes créés

Première connexions

De la même manière que dans [5], appelons τ_1 la durée qui s'est écoulée entre la création d'un compte et la première tentative de connexion à ce compte réussie et τ_2 la durée écoulée depuis cette tentative jusqu'à la première connexion avec saisie de commandes sur ce compte. Les résultats du calcul de τ_1 et τ_2 pour l'ensemble des comptes créés sur les pots de miel sont donnés dans la table 5.5. Par ailleurs, sur certains pots de miel, un attaquant s'étant connecté sur un des 11 comptes créés a réussi à obtenir les droits root en exploitant une faille du système d'exploitation. Quand cela a été le cas, nous donnons le temps qui s'est écoulé entre la création des comptes et cet évènement.

Compte	Toulouse		Rennes		C.P.		LAAS	
	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2
C1	65h	-	48h	81h	-	-	25j	-
C2	61h	-	48h	15j	-	-	14j	-
C3	32h	-	72h	15j	-	-	33h	37h
C4	-	-	7j	14j	-	-	-	-
C5	65h	-	72h	99h	-	-	6j	-
C6	65h	70h	64h	7h	-	-	6j	11j
C7	50h	1h	72h	6j	-	-	55h	1h
C8	7j	19h	99h	99h	-	-	100h	10j
C9	65h	70h	72h	28h	23h	6h	60h	2h
C10	56h	-	72h	9j	-	-	55h	1h
C11	65h	-	72h	16j	-	-	7j	6j
root	6j		-		-		25j	

TABLE 5.5 – Temps entre les premières connexions en fonction du couple et du pot de miel ciblé

Tout d'abord, le tableau montre qu'il n'y a eu qu'un compte trouvé sur le pot de miel de College Park. Il y a plusieurs raisons possibles à cela. En premier lieu, l'activité sur ce pot de miel depuis la création des comptes a été bien plus faible que sur les trois autres. Cela aurait pu être dû à une configuration des paramètres du réseau, qui pourrait par exemple rejeter les connexions générant un trafic trop important (on parle de *rate limiting*) comme dans le cas d'une attaque par dictionnaire. Différents tests indiquent que ce n'est pas le cas. Une autre hypothèse concerne le délai de quelques secondes induit par la redirection des connexions depuis les États-Unis vers la France. Ce délai pourrait être assez important pour décourager une partie des attaquants. Enfin, il n'est pas exclu qu'un attaquant ait trouvé un moyen de démasquer le pot de miel, mettant alors l'adresse IP correspondante sur une « liste noire »des machines à ne pas attaquer.

Nous voyons sur ce tableau que tous les comptes ayant été trouvés (τ_1 est fini) n'ont pas nécessairement été attaqués par la suite. Cela peut être dû à plusieurs raisons. Tout d'abord, comme nous avons délibérément choisi des couples fréquemment tentés, il arrive que lors d'une même session d'attaque plusieurs couples soient trouvés. Il est arrivé que l'attaquant ne se connecte que sur un de ces comptes et modifie alors le mot de passe des autres depuis celui-ci, empêchant d'autres attaquants de s'y connecter à leur tour. De plus, nous avons eu plusieurs cas où un attaquant a réussi à prendre le contrôle du compte root. Ici aussi, il a ensuite modifié les mots de passe des comptes qu'il a détectés sur la machine. Étant devenu root, il possédait tous les droits sur la machine attaquée et n'avait même plus besoin de connaître le mot de passe original du compte à modifier. Dès lors, ces nouveaux mots de passe étant relativement complexes, ils ne seront pas trouvés lors de futures attaques par dictionnaire. Ce genre de comportements spécifiques sera détaillé dans la partie 5.3.2.

Concernant les résultats obtenus, on constate que la découverte des comptes s'est faite rapidement, comme prévu, puisqu'en une semaine tous les identifiants et mots de passe sauf un ont été découverts. En revanche, le temps avant une première intrusion sur ces comptes est bien plus variable, allant d'une heure à plus de deux semaines.

Répartition géographique

Les connexions enregistrées provenaient de 501 adresses IP distinctes, issues de 62 pays. Le tableau 5.6 donne les 5 pays pour lesquels le nombre d'adresses IP différentes ayant été aperçues sur chaque pot de miel est le plus important.

Classement	Toulouse	Rennes	College Park	LAAS
1	Chine 40	Chine 34	Chine 14	Chine 45
2	États-Unis 23	États-Unis 20	États-Unis 13	États-Unis 23
3	Roumanie 11	Corée du Sud 12	Argentine 5	Roumanie 10
4	Corée du Sud 10	Roumanie 9	Brésil 4	France 8
5	Russie 8	Allemagne 6	Thailande/Taiwan 3	Russie 8
Total IP différentes	197	153	63	181

TABLE 5.6 – Pays d'origine des attaques les plus observés sur chaque pot de miel

D'après le tableau 5.6, la Chine et les États-Unis se démarquent nettement, et ce sur les quatre pots de miel. Ensuite, nous observons que les mêmes pays apparaissent sur les trois pots de miel situés en France alors qu'ils sont absents sur celui des États-Unis. Cependant, les écarts ne sont pas suffisamment significatifs au bout de deux mois d'observation pour pouvoir conclure.

Par ailleurs, en comparant les IP collectées par ces quatre pots de miel avec celles collectées par le pot de miel précédemment déployé au LAAS (3230 adresses différentes répertoriées entre janvier 2006 et août 2010), seulement 3 d'entre elles ont été vues lors des deux expériences et ce uniquement sur les pots de miel situés en France. Par conséquent, il semble que les machines, ou du moins les adresses IP, utilisées pour commettre ces attaques ne le soient que pendant une durée limitée.

5.1.2 Identification des sessions

Les analyses des données dans leur ensemble nous permettent d'obtenir des informations générales à propos du comportement des attaquants et de leurs outils. Cependant, cela ne nous permet pas de comparer les différents comportements des attaquants. Pour ce faire, il est nécessaire d'isoler les activités correspondant à chacune des attaques. Nous avons décidé de regrouper les connexions en différentes sessions, selon la méthode décrite en 4.2.3. Cependant, nous avons ici procédé en trois étapes afin de déterminer le seuil nécessaire à la séparation des sessions. Tout

d'abord, nous avons lancé des attaques pour identifier l'influence du type d'algorithme sur les observations. Cette partie englobe uniquement des analyses qualitatives. Ensuite, nous avons fixé le seuil à partir des données observées. Pour finir, nous avons réalisé le regroupement en sessions. Avant d'« ouvrir »nos pots de miel au monde entier, nous les avons soumis pendant une journée à des attaques par dictionnaires en utilisant deux programmes différents. Ces programmes ont été lancés plusieurs fois pendant quinze minutes sur chacun des pots de miel. Deux exécutions successives ont été espacées d'au moins cinq minutes afin de pouvoir les distinguer par la suite.

- `THC Hydra`[1] est un programme en C permettant de lancer des attaques de type *bruteforce* pour divers protocoles, dont ssh. Le programme prend en paramètre une liste d'adresses cibles, une liste d'identifiants et une liste de mots de passe. Il va alors tenter des combinaisons identifiant/mot de passe (par défaut, il parcourt les listes dans l'ordre) sur chacun des hôtes de la liste des cibles. Il possède également un générateur permettant de créer des mots à tenter selon certains paramètres (nombre de caractères, présence ou non de chiffres, caractères imposés ou interdits, etc.). Enfin, il peut lancer plusieurs processus pour effectuer plusieurs tentatives simultanément.
- `Brutessh`[2], écrit en Python, prend en paramètres l'adresse de l'hôte à attaquer, un identifiant et une liste de mot de passe. Il teste ensuite cette liste dans l'ordre sur l'hôte donné. Il ilance également plusieurs processus (12 par défaut) afin de paralléliser les tentatives de connexion.

Nous avons alors compté les temps entre deux tentatives de connexion successives enregistrées par les pots de miel. D'après nos mesures, l'immense majorité des connexions sont séparées de moins d'une seconde. En nous basant uniquement sur ce résultat, une valeur seuil de 1 seconde nous permettrait théoriquement de reconstituer la majeure partie des sessions. Cependant, il arrive que deux tentatives successives, pourtant lancées lors d'une même attaque, soient séparées par une durée supérieure à ce seuil. Ces quantités en apparence négligeables (moins de 0.01%) ont en fait un grand impact sur notre séparation. Ainsi, lors de nos tests, un tel intervalle s'est produit en moyenne (sur une durée de quinze minutes) 78 fois lors de l'utilisation de `Hydra` et 18 fois avec `Brutessh`. Deux conclusions s'imposent alors :

- Il n'est pas possible de fixer un seuil convenable avec cette définition, puisque par exemple cela pourrait nous séparer une unique attaque lancée avec brutessh en 18 sessions.
- Nous voyons également que l'algorithme utilisé pour de telles attaques influence fortement nos observations. En effet, malgré des interruptions plus nombreuses et plus longues, `Hydra` effectue environ 55% de tentatives en plus par rapport à `Brutessh`. En prenant également en compte les grandes variations de la puissance de calcul selon les machines, nous devrons donc fixer le seuil en nous basant sur l'ensemble des résultats collectés.

Sur la base des données collectées, nous avons fait varier la valeur du seuil entre 2 et 200 secondes et calculé le nombre de sessions obtenues à chaque fois. Les résultats sont visibles dans la figure 5.1. Pour déterminer le seuil, nous avons alors regardé le pourcentage de diminution du nombre de sessions entre deux valeurs successives : pour chaque valeur n du seuil, si $S(n)$ est le nombre de sessions obtenues pour ce seuil, nous avons calculé la valeur $\frac{S(n)-S(n+1)}{S(n)}$ dont on peut voir les résultats exprimés en pourcentages sur la figure 5.2. Ce nombre correspond au pourcentage de regroupement supplémentaire si on augmente la valeur du seuil de une seconde. Il apparait alors trois phases :

- Tout d'abord, la décroissance est très forte, le fait d'augmenter le seuil d'une seconde entraîne une chute de plus de 10% de cette quantité. Cette phase concerne les 7 premières secondes.
- Ensuite, la chute est plus faible et se situe autour des 5%.

1. http ://thc.org/thc-hydra/
2. http ://www.edge-security.com/edge-soft.php

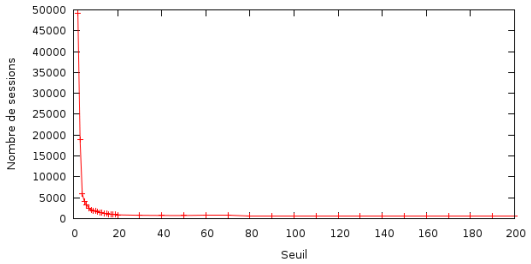

FIGURE 5.1 – Évolution du nombre de sessions en fonction du seuil

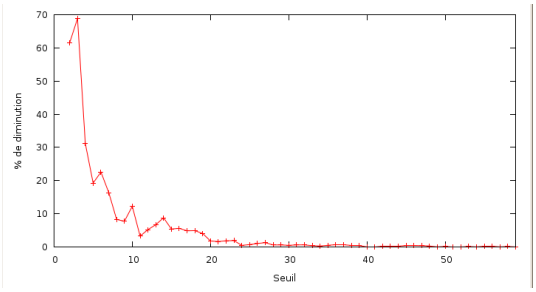

FIGURE 5.2 – Décroissance du nombre de sessions en fonction du seuil

– Enfin, pour des valeurs supérieures à 20 secondes la diminution devient inférieure ou égale à 1% et tend vers 0.

Nous avons alors fixé empiriquement la valeur du seuil à 20 secondes, ce qui correspond à la fin de la deuxième phase. En effet, après celle-ci, il y a peu de variations dans la quantité de sessions trouvées et l'on risquera plus de regrouper deux attaques en réalité séparées (20 secondes sont amplement suffisantes pour interrompre un programme et le relancer avec de nouveaux paramètres) que de casser en plusieurs morceaux une unique attaque.

Caractérisation des attaques

Concernant la caractérisation des sessions ainsi constituées, nous conservons la classification établie dans [5]. Nous distinguons ainsi deux catégories rappelées dans 3.2 : attaques par dictionnaire et intrusions. Les sessions pour lesquelles nous ne pouvons pas conclure (pas de commandes saisies et très peu de couples tentés) sont regroupées dans une troisième catégorie. Ainsi, après deux mois (dont un sans comptes créées, et donc sans intrusions possibles) la répartition selon ces trois catégories est représentée dans le tableau 5.7.

	Dictionnaire	Intrusion	Autre	Total
Toulouse	170	21	91	282
Rennes	134	23	43	200
College Park	63	1	71	135
LAAS	153	24	151	328
Total	520	69	356	945

TABLE 5.7 – Répartition des sessions sur chaque pot de miel

Il y a ainsi eu 945 sessions d'attaques observées sur l'ensemble des pots de miel sur une période de deux mois. Notons cependant que le grand nombre d'attaques tombant dans la catégorie « Autre »au LAAS est en grande partie dû à mise en place à notre insu d'un contrôle des connexions entrantes sur le réseau. Ainsi, lorsque trop de requêtes parvenaient d'une même source très rapprochées dans le temps, la connexion avec notre pot de miel était automatiquement interrompue.

Par ailleurs, en faisant l'intersection de l'ensemble des adresses utilisées lors d'attaques par dictionnaire avec l'ensemble des adresses utilisées lors d'intrusion, nous obtenons un ensemble vide. Ceci est vrai au sein de chaque pot de miel ainsi qu'en les considérant tous ensemble. Cela confirme les résultats obtenus lors des expériences précédentes : les machines utilisées lors d'une attaque sont spécialisées pour effectuer des tâches bien précises.

Nous allons maintenant détailler les analyses effectuées et les résultats obtenus sur les attaques par dictionnaire et les intrusions.

5.2 Attaques par dictionnaire

5.2.1 Généralités

Nous appelons attaque par dictionnaire une session pendant laquelle au moins 9 tentatives de connexion par ssh ont eu lieu. Cela nous permet d'éliminer de manière relativement sure les cas où une attaque s'avèrerait en fait être une erreur de connexion. Parmi les sessions obtenues avec un seuil de 20s, nous obtenons un total de 520 attaques par dictionnaire. Celles-ci ont été lancées depuis 329 adresses distinctes, provenant de 58 pays différents.

5.2.2 Vocabulaires

Définition

Nous appelons vocabulaire d'une session l'ensemble des couples identifiant/mot de passe utilisés lors de cette session. Chaque couple constitue un mot de ce vocabulaire. Nous appelons dictionnaire un regroupement de vocabulaires selon des paramètres communs.

Observations

Nous avons créé un dictionnaire général pour chaque pot de miel qui est en fait l'union de tous les vocabulaires observés sur ce pot de miel. Nous appelons ainsi $D1, D2, D3$ et $D4$ les dictionnaires généraux des pots de miel respectivement situés à Toulouse, Rennes, College Park et au LAAS. Nous appelons $D1', D2', D3'$ et $D4'$ les dictionnaires créés à partir de D1, D2, D3 et D4 contenant les mots apparaissant uniquement dans ces dictionnaires :

$$Di' = \{m \in Di / \forall j \neq i, m \notin Dj\}$$

Ce sont les parts exclusives de ces dictionnaires. Le tableau 5.8 montre ainsi quelles quantités de mots ces dictionnaires ont en commun.

Malgré les fortes différences de taille entre les différents dictionnaires, nous pouvons tout de même faire quelques observations. Tout d'abord, il est clair qu'il existe une base de couples communs répandue dans de nombreux dictionnaires. En effet, $D2$, $D3$ et $D4$, c'est à dire les dictionnaires n'étant pas les plus fournis, ont une part exclusive qui compte respectivement pour 12%, 20% et 10% de leur taille totale.

De la même manière, nous construisons $D0$, dictionnaire contenant tous les couples tentés lors du déploiement des années précédentes et $D0'$, sa part exclusive par rapport à $D1, D2, D3$ et $D4$. Le tableau 5.9 nous permet ainsi d'avoir un aperçu de l'évolution des vocabulaires actuels par rapport à ceux observés lors des années précédentes. Ces résultats confirment l'existence de dictionnaires largement utilisés mais ne semblent pas indiquer de forte évolution de ceux-ci au cours des années. Par exemple, au moins 50% du contenu des dictionnaires observés récemment se retrouve dans $D0$.

Concernant les tailles absolues des différents ensembles, les fortes disparités entre les quantités de données enregistrées par les pots de miel ne nous permettent pas de tirer des conclusions avec certitude. Cependant nous constatons alors que pour tous les dictionnaires considérés, si nous notons $|A|$ le nombre de mots du dictionnaire A, la règle suivante s'applique :

$$\forall A, B, C, |B| > |C| \Leftrightarrow |A \cap B| > |A \cap C|$$

En l'état actuel, cela tend à montrer une homogénéité dans la répartition des couples utilisés sur chaque pot de miel. En d'autres termes, les disparités entre les dictionnaires généraux de chaque pot de miel semblent surtout dues aux grands écarts entre les quantités de données récoltées par ces pots de miel. Ainsi, en continuant de collecter des données sur plusieurs mois puis en

Dictionnaire	Nombre de mots
$D1$	96735
$D2$	41789
$D3$	4526
$D4$	6083
$D1 \cap D2$	26091
$D1 \cap D3$	2429
$D1 \cap D4$	5182
$D2 \cap D3$	2783
$D2 \cap D4$	3885
$D3 \cap D4$	1013
$D1 \cap D2 \cap D3$	1934
$D1 \cap D2 \cap D4$	3762
$D1 \cap D3 \cap D4$	935
$D2 \cap D3 \cap D4$	882
$D1 \cap D2 \cap D3 \cap D4$	864
$D1'$	68710
$D2'$	5163
$D3'$	918
$D4'$	628

TABLE 5.8 – Intersections des différents dictionnaires généraux

Dictionnaire	Nombre de mots
$D0$	253287
$D0 \cap D1$	45738
$D0 \cap D2$	28831
$D0 \cap D3$	3441
$D0 \cap D4$	4622
$D0 \cap D1 \cap D2 \cap D3 \cap D4$	840
$D0'$	196646

TABLE 5.9 – Comparaisons entre ancien et nouveaux dictionnaires

refaisant cette analyse, nous devrions trouver des intersections de dictionnaires bien plus fournies proportionnellement à la taille des ensembles considérés. Il serait cependant intéressant de faire des regroupements plus fins de ces vocabulaires, d'abord au sein de chaque pot de miel puis sur leurs intersections afin d'observer des tendances locales dans le choix des dictionnaires utilisés par les attaquants en un moment donné. Cela fait partie des travaux mentionnés en partie 5.4.

5.2.3 Répartition géographique

Origine des attaques

Le tableau 5.10 donne les 5 pays les plus impliqués dans les attaques par dictionnaire de nos pots de miel sachant que nous comptabilisons chaque adresse IP une seule fois. En comparant ces chiffres avec le tableau 5.6, il apparait que les adresses IP chinoises et américaines sont majoritairement utilisées pour mener des attaques par dictionnaire. Globalement, en plus de ces deux pays, d'autres sont présents dans le haut du classement sur les trois machines situées en France, comme la Corée du Sud, le Canada ou la Russie. En revanche, ils n'apparaissent pas dans les premières positions à College Park. Les machines situées dans ces régions (ou du moins utilisant des adresses IP situées dans ces régions) semblent donc dédiées à l'attaque d'une plage d'adresses incluant nos trois pots de miel français.

Classement	Toulouse		Rennes		College Park		LAAS	
1	Chine	32	Chine	25	Etats-Unis	8	Chine	28
2	États-Unis	20	États-Unis	14	Chine	4	États-Unis	11
3	Russie	8	Corée du Sud	11	Argentine	4	France	5
4	Corée du Sud	8	Turquie	5	Taiwan	2	Canada	5
5	Inde	6	Canada	5	Thailande	2	Russie	4

TABLE 5.10 – Pays les plus observés sur chaque pot de miel pour des attaques par dictionnaire

Cibles des attaques par dictionnaire

Le tableau 5.11 présente les quantités d'adresses IP ayant effectué des attaques par dictionnaire sur plusieurs machines. Nous constatons ainsi qu'aucune adresse n'a attaqué les quatre pots de miel.
En revanche, il y a un nombre important d'adresses ayant visité au moins deux pots de miel en France mais au contraire très peu d'adresses observées en France et aux États-Unis. Or, les adresses IP correspondant aux pots de miel français sont assez rapprochées entre elles (elles commencent par 192, 193 et 195) mais plus éloignées de celle de College Park (commençant par 128). De plus, les séquences de couples testés par une même adresse sur plusieurs de nos pots de miel sont très similaires, voire identiques. Il semble donc que des machines dédiées à ces attaques se voient attribuer des plages d'IP à attaquer plutôt que de balayer tout le spectre possible et se contentent de répéter la même séquence de couples à tester sur chaque machine avant de passer à la suivante, voire de recommencer cette séquence sur toute la plage, comme nous l'avons observé plusieurs fois.

Ensemble	Nombre d'attaques	IP distinctes
Toulouse	170	148
Rennes	134	109
College Park	63	29
LAAS	153	95
Toulouse ∩ Rennes	65	31
Toulouse ∩ College Park	1	1
Toulouse ∩ LAAS	65	36
Rennes ∩ College Park	0	0
Rennes ∩ LAAS	51	25
College Park ∩ LAAS	3	2
Toulouse ∩ Rennes ∩ LAAS	53	18
Toulouse ∩ College Park ∩ LAAS	0	0

TABLE 5.11 – Répartition des adresses IP sources des attaques par dictionnaire

5.3 Intrusions

Dans cette section, nous nous intéressons aux cas où des connexions réussies par les attaquants ont été suivies par une saisie de commandes de leur part.

5.3.1 Identification des attaquants

Tout d'abord, observons d'où proviennent ces intrusions. D'après le tableau 5.12, nous voyons une très forte présence d'adresses IP provenant de Roumanie (25 au total), secondée par l'Italie (6 au total). Il est toutefois possible que certaines de ces adresses ne soient en fait que des relais utilisés par l'attaquant pour brouiller les pistes. Cependant, l'analyse des données fournies par les terminaux utilisés par les attaquants a montré qu'ils tentaient souvent de télécharger

des programmes sur des sites hébergés en Roumanie, mais surtout que parmi les programmes effectivement exécutés, plusieurs affichaient du texte en Roumain.

Classement	Toulouse		Rennes		College Park		LAAS	
1	Roumanie	8	Roumanie	9	Italie	1	Roumanie	8
2	Italie	2	USA/UK/Liban	2	-		Italie	3
3	Suède/Allemagne	1	Mexique/Italie/Espagne	1	-		Russie	2
4	-		-		-		Moldavie	1

TABLE 5.12 – Pays observés sur chaque pot de miel pour des intrusions

De plus, nous avons vu dans la partie 5.1.1 que des adresses IP avaient été observées sur plusieurs machines. Les résultats de l'analyse des recoupements d'adresses visibles dans le tableau 5.13 nous montrent que les intrusions sont rarement menées depuis les mêmes adresses. En effet, seules quatre adresses IP ayant effectué des intrusions ont été vues sur plusieurs pots de miel, plus spécifiquement sur ceux situés à Toulouse et Rennes. En observant plus en détail le contenu des intrusions réalisées par ces quatre adresses, il apparait de plus que le mot de passe modifié utilisé lors de l'appropriation du compte (cf 5.3.2) est toujours le même. Ces quatre adresses sont donc possédées par le même individu ou le même groupe d'individus. Par ailleurs, le nombre d'adresses distinctes est relativement proche du nombre total d'intrusions répertoriées mais bien supérieur au nombre de comptes attaqués, ce qui signifie que dans la majorité des cas les attaquants ne se connectent pas plusieurs fois (du moins avec la même adresse) sur un compte. Dans le cas des comptes ayant été visités par plusieurs adresses, deux hypothèses (ne s'excluant pas mutuellement) sont alors envisageables :

- L'attaquant cherche à brouiller les pistes pour qu'on ne puisse pas facilement remonter jusqu'à lui, et change donc d'adresse à chaque connexion.
- Il y a en fait une communauté d'attaquants se partageant des informations à propos des machines qu'ils ont attaquées, ce qui leur permet par exemple de se relayer si l'un d'entre eux ne parvient pas à prendre le contrôle d'une machine par manque de connaissances techniques.

Ensemble	Nombre d'intrusions	IP distinctes	Nb. comptes attaqués
Toulouse	21	12	4 + root
Rennes	23	18	11
College Park	1	1	1
LAAS	24	14	7 + root
Toulouse ∩ Rennes	11	4	2 + 2
Toulouse ∩ College Park	0	0	-
Toulouse ∩ LAAS	0	0	-
Rennes ∩ College Park	0	0	-
Rennes ∩ LAAS	0	0	-
College Park ∩ LAAS	0	0	-

TABLE 5.13 – Répartition des adresses IP sources des intrusions

5.3.2 Activités des attaquants

Tendances globales

Lors des 69 intrusions observées, plusieurs comportements communs à un grand nombre d'attaquants ont été régulièrement observés :

- Souci de discrétion : les attaquants commencent par regarder si ils sont les seuls connectés sur la machine, par exemple avec la commande `w`. Un certain nombre d'entre eux efface également des fichiers d'historique avant de se déconnecter.

- Prise de connaissance de la machine : Les attaquants vont chercher à récupérer des informations sur l'architecture de la machine attaquée : nom et version de l'OS, caractéristique du processeur, etc. Ces informations vont leur permettre de déterminer comment mener leur attaque (par exemple, la version du noyau peut donner une indication sur les failles de sécurité qui ne seront pas exploitables).
- Appropriation du compte : Lors de leur première connexion à un compte, les attaquants changent toujours le mot de passe de celui ci par quelque chose de bien plus complexe afin de s'approprier le compte, prenant ainsi le risque de se faire détecter lorsque l'utilisateur légitime voudra utiliser sa machine. Nous avons observé un seul cas où l'attaquant a remis le mot de passe original avant de se déconnecter.
- Scan d'adresses IP : l'attaquant installe un programme qui va scanner une plage d'adresses IP afin de voir lesquelles sont accessibles depuis la machine attaquée. Le but ici est d'utiliser cette dernière comme une passerelle pour mener de nouvelles attaques. Au vu des paramètres de sécurité de notre infrastructure, de tels scans ont toujours échoué.
- Installation de client `IRC` : Ce client de messagerie est en fait un *bot*, c'est à dire un programme automatisé utilisant le protocole IRC, qui est employé de manière malveillante pour recevoir et exécuter des instructions envoyées par un serveur distant. L'objectif ici semble être de rattacher la machine piratée à un *botnet*, le serveur envoyant alors ses ordres à des centaines de machines en même temps.
- Tentatives d'accession aux privilèges d'administrateur : Certains attaquants cherchent à obtenir les privilèges d'administrateur afin d'obtenir un contrôle total sur la machine piratée. Pour cela, ils vont chercher à exploiter des failles de sécurité de l'OS via divers programmes spécialisés dans l'exploitation d'une faille particulière.

Nous n'avons pas constaté de différences importantes de comportement selon les sites attaqués. De même, il n'y a pas eu de tentative particulière en fonction de l'identifiant du compte utilisé. Par exemple, aucun attaquant connecté sur le compte « apache »n'a essayé de voir si un serveur web s'exécutait ni si des données étaient accessibles.

Pleins pouvoirs sur le système

Une fois les privilèges d'administrateur obtenus, les attaquants ont aussitôt changé le mot de passe root de la machine. Ils ont ensuite utilisé ces privilèges pour installer des programmes modifiés afin d'obtenir des informations sur les utilisateurs « légitimes » de la machine ainsi que pour ouvrir un nouveau port afin de pouvoir continuer à communiquer avec la machine s'il venaient à perdre l'accès via le port 22 (on parle de *porte dérobée*).

- Le premier a ainsi installé le `rootkit` SHV4[3], qui installe un serveur `ssh` si il n'y en a pas déjà un présent, modifie l'exécutable du client `ssh` pour qu'il enregistre les couples identifiant/mot de passe tentés lors de connexions vers d'autres hôtes et surtout installe un grand nombre de versions modifiées d'exécutables du système qui pourraient permettre de détecter sa présence afin de rester indétectable. Cet attaquant cherche donc vraisemblablement à obtenir facilement de nouvelles cibles pour de futures attaques. Cependant, il a également modifié le mot de passe de tous les comptes qu'il a pu trouver sur la machine grâce à son attaque par dictionnaire précédent l'intrusion, ainsi qu'en listant les dossiers du répertoire `/home` (mais sans consulter la liste des comptes existants, par exemple dans `/etc/passwd`). Il semble ainsi étrange que l'attaquant cherche à couvrir ses traces et à récupérer les mots de passe fournis par les personnes pouvant utiliser la machine tout en bloquant l'accès au plus grand nombre de comptes possible.

3. Une analyse plus détaillée peut être vue sur http ://web.fhnw.ch/plattformen/ns/vorlesungsunterlagen-1/network-analysis-tools/shv4-analysis

– Le second a également remplacé le binaire du client `ssh` par une autre version, probablement dans le même objectif, mais nous n'avons pas de certitude quant aux modifications réellement apportées. Il n'a en revanche pas changé les mots de passe d'autres comptes existants, mais en a créé un nouveau, nommé « backup »et possédant les droits d'administrateur. Il s'est depuis reconnecté via ce compte mais n'a pour l'instant rien modifié de plus. En revanche, il continue d'utiliser le compte « user »pour installer de nouveaux clients `IRC`, probablement car établir un contact avec ceux précédemment installés lui est impossible du fait de notre verrouillage du réseau.

5.4 Travaux en cours

Le découpage en sessions et l'analyse de celles-ci nous a permis d'obtenir de nouvelles informations pour classifier les attaquants. Cependant, il est évident que cette méthode n'est pas l'unique moyen possible pour regrouper les données . Actuellement, nous sommes en train de chercher de nouveaux moyens pour regrouper et classifier ces données.

Par exemple, d'une manière similaire à celle utilisée lors des comparaisons des ensembles de combinaisons tentées sur chaque pot de miel, nous travaillons sur un regroupement des vocabulaires utilisés lors de chaque session d'attaque par dictionnaire en nous inspirant du travail effectué dans [5]. Là aussi, l'accès aux données enregistrées lors des cinq années d'expérimentation précédentes nous sera utile pour constater une éventuelle évolution de ces dictionnaires (regroupement, apparition ou disparitions de certains d'entre eux par exemple).

Dans le même ordre d'idée, nous sommes également en train d'essayer de créer des vocabulaires d'intrusion, consistant en l'ensemble des commandes tapées lors d'une même intrusion. Ce vocabulaire doit tenir compte des noms des programmes et de leurs paramètres qui leurs sont passés. La difficulté supplémentaire par rapport aux vocabulaires d'attaque est que toute ligne rentrée par l'attaquant n'est pas nécessairement une instruction. En effet, il faut tenir compte des fautes de frappe qu'un attaquant peut faire, rendant alors son instruction invalide. Il faut également gérer les moments où l'attaquant fait appel à un éditeur de texte (tel que `nano` ou `vi`), les raccourcis claviers qu'il peut utiliser, etc.

De plus, à l'heure actuelle toutes les analyses ont été effectuées grâce à des programmes et scripts développés par nos soins, en fonction des besoins découlant des observations faites directement sur la base de donnée ou via l'interface détaillée en 4.2.3. Il serait intéressant d'utiliser un logiciel de *data mining* afin d'explorer la base et trouver des associations auxquelles nous n'avons pas pensé.

Enfin, nous sommes arrivés au terme du deuxième mois de l'expérience, et les activités sur les pots de miel semblent s'être tassées, en particulier en ce qui concerne les intrusions (une grande majorité des comptes ayant été utilisée et modifiée), et les deux machines virtuelles sur lesquelles le compte root a été attaqué avec succès possèdent des programmes corrompus. Par conséquent, il va être temps de remettre ces pots de miel dans leur état initial et de relancer l'expérience d'ici la fin du mois d'août avec de nouveaux paramètres, par exemple avec une liste de comptes différents ou un noyau sur lequel les vulnérabilités exploitées pour passer root ne fonctionnent plus.

Chapitre 6

Conclusion

Comme montré dans notre état de l'art des pots de miel à travers une liste non exhaustive d'exemples, les applications et axes de développement possibles des pots de miel sont très vastes. Lors de ce stage, nous avons pu contribuer à ces développements en améliorant le pot de miel développé au LAAS grâce à la conception d'une architecture réseau permettant un déploiement aisé de celui-ci en divers sites à travers le monde. Grâce à l'emploi de tunnels `GRE` et la mise en place de règles de routage des paquets pour rediriger les connexions vers une machine virtuelle située au LAAS, il devient aisé de déployer notre pot de miel en de nouveaux sites pour diversifier encore nos sources de données.

De plus, la conception d'un système de stockage via une base de données ainsi que le développement de scripts automatisant la collecte et le traitement de ces données nous permettent un accès rapide à l'ensemble de celles-ci. Enfin, l'interface de visualisation des données, en facilitant la lecture et les interactions avec la base, a fourni un gain de temps précieux pour mener les analyses dont les premiers résultats sont présentés dans ce rapport.

Malgré quelques soucis d'ordre technique, ces résultats sont conséquents et permettent déjà de dénoter une homogénéité aussi bien spatiale (sur deux continents) que temporelle (sur cinq ans) dans le comportement des attaquants à travers le monde. Il apparaît que ceux-ci opèrent maintenant toujours dans le but de constituer des *botnets* et n'hésitent pas à corrompre complètement une machine pour atteindre leur objectif.

À l'heure actuelle, en plus des travaux en cours sur la constitution de vocabulaires d'attaque et la confrontation des vocabulaires de sessions, de nombreux travaux sont possibles. Concernant la collecte des données, il faudra obtenir plus de partenariats permettant de déployer le pot de miel sur de nouveaux sites, de préférence hors de France. Cela permettra de confirmer les hypothèses faites d'après les analyses des quatre pots de miel actuel. De plus, posséder plusieurs pots de miel pour chaque zone géographique (contrairement à notre cas où nous en avions trois en France, mais un seul aux États-Unis) minimiserait les problèmes dûs à une faible activité sur un pot de miel, comme cela a été le cas à College Park. Par ailleurs, notre système permettant un remplacement aisé des pots de miel (il suffit de changer l'image de la machine virtuelle, et éventuellement de modifier les règles de routage ou de filtrage), il sera intéressant d'implémenter de nouvelles vulnérabilités à ce pot de miel puis de le déployer sur tous les sites.

Glossaire

Apache Logiciel libre de serveur HTTP très répandu. Il fonctionne aussi bien sur les systèmes d'exploitations UNIX que sous Windows.

Bash *Bourne Again Shell.* Interface en ligne de commande permettant d'interagir avec un système d'exploitation UNIX.

Bot En informatique, logiciel automatique qui interagit avec un serveur distant. Ils permettent d'effectuer rapidement des tâches répétitives lorsqu'ils en reçoivent l'instruction.

Botnet Ensemble de bots connectés entre eux. Dans le cas de machines infectées par un bot malveillant, cela permet au pirate de se constituer un réseau de « machines zombies », c'est à dire des ordinateurs exécutant simultanément des instructions, envoyées par celui qui les a piratés, à l'insu de leur propriétaire légitime. Les botnets sont par exemple employés pour envoyer massivement des pourriels ou surcharger un site web de requêtes pour l'empêcher de fonctionner correctement (attaque par déni de service distribué).

Cheval de Troie Logiciel malveillant camouflé en programme légitime. Grâce aux autorisations alors données par l'utilisateur croyant utiliser un programme sans danger, un cheval de Troie pourra alors accéder à des données confidentielles ou installer une porte dérobée sur l'ordinateur hôte.

Clustering Utilisé dans le sens de *data clustering*, partitionnement de données. C'est une méthode d'analyse permettant de regrouper entre elles des données similaires selon des critères prédéfinis.

***Data mining* (Fouille de données)** Ensemble de techniques permettant d'extraire des informations intéressantes à partir d'un grand nombre de données.

DNS *Domain Name System.* Protocole permettant de faire le lien entre un nom de domaine et l'adresse IP correspondante.

FTP *File Transfer Protocol.* Protocole de transfert de fichiers selon un système client-serveur.

ICMP *Internet Control Message Protocol.* Protocole par lequel transitent les messages d'erreur ou de contrôle utilisés par la suite de protocoles Internet. Par exemple, la commande `ping` utilise ce protocole pour savoir si une machine est accessible sur le réseau.

IRC *Internet Relay Chat*. Protocole de messagerie sur Internet fonctionnant sur un système client-serveur. Il est utilisé principalement pour la communication instantanée et les discussions en groupe mais permet également le transfert de fichiers.

Machine virtuelle Émulation logicielle d'un appareil informatique. Cette émulation se fait grâce à un logiciel qui simulera le matériel et les logiciels nécessaires au fonctionnement d'une machine réelle équivalente.

MySQL Système de gestion de base de données (SGBD). C'est un programme permettant de créer et manipuler des bases de données.

NTP *Network Time Protocol*. Protocole permettant de synchroniser l'horloge d'un ordinateur via le réseau.

Rootkit Ensemble d'outils logiciels dont le but est de fournir un accès à un ordinateur dissimulé à son utilisateur légitime. Un rootkit est très souvent utilisé dans un but malveillant.

OS *Operating System*. Système d'exploitation d'un ordinateur, tel que Linux ou Windows.

SSH *Secure Shell*. Protocole de communication sécurisé entre deux ordinateurs, permettant entre autres d'obtenir un terminal en lignes de commande sur une machine distante. Les communications sont chiffrées, empêchant donc un tiers de connaitre leur contenu.

TCP *Transmission Control Protocol*. Protocole de transport en mode connecté utilisé par Internet.

TTY Vient historiquement de l'anglais *TeleTYpewriter* (télescripteur). Aujourd'hui, il désigne une fenêtre contenant un terminal de commandes.

UDP *User Datagram Protocol*. Protocole de transport en mode non-connecté utilisé par Internet.

Ver En informatique, logiciel malveillant se propageant automatiquement d'une machine à une autre via le réseau.

VPN *Virtual Private Network*, Réseau Privé Virtuel. Connexion entre des réseaux locaux à travers un autre réseau (typiquement, Internet) via l'utilisation de tunnels. Ces réseaux locaux peuvent alors communiquer comme si il existait une connexion physique directe entre eux.

Bibliographie

[1] McCarty (B.), « The honeynet arms race », *IEEE Security and Privacy*, vol. 1, 2003, p. 79–82.

[2] Cheswick (B.), « An evening with berferd in which a cracker is lured, endured, and studied », dans *Proceedings of the Winter 1992 USENIX Conference*, p. 163–174, 1992.

[3] Maggi (F.) et Zanero (S.), « Analysis of the state of the art », *WOMBAT Project Worldwide Observatory of Malicious Behaviors and Attack Threats*, 2008. http ://wombat-project.eu/workpackages/wp2-analysis-of-state-of-the-a/.

[4] NOAH, « Do1.1 : Survey on the state of the art », *Deliverable of the European Network of Affined Honeypots*, 2005. http ://www.fp6-noah.org/publications/deliverables/D0.1.pdf.

[5] Alata (E.), *Observation, caractérisation et modélisation de processus d'attaques sur Internet.* Thèse de doctorat, Institut National des Sciences Appliquées de Toulouse, 2007.

[6] Pouget (F.) et Holz (T.), « A pointillist approach for comparing honeypots », *Intrusion and Malware Detection and Vulnerability Assessment*, 2005, p. 51–68.

[7] Provos (N.), « A virtual honeypot framework », *Proceedings of the 13th conference on USENIX Security Symposium*, 2004.

[8] Balas (E.), « Know your enemy : Sebek », *The Honeynet Project*, 2003. www.honeynet.org/papers/.

[9] Dornseif (M.), Holz (T.) et Klein (C.), « Nosebreak - attacking honeynets », dans *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, p. 123–129, juin 2004.

[10] Leita (C.), Mermoud (K.) et Dacier (M.), « Scriptgen : an automated script generation tool for honeyd », dans *Proceedings of the 21st Annual Computer Security Applications Conference*. IEEE, 2005.

[11] Wagener (G.), State (R.), Dulaunoy (A.) et Engel (T.), « Self adaptive high interaction honeypots driven by game theory », dans Guerraoui (R.) et Petit (F.), éditeurs, *SSS*, vol. 5873 (coll. *Lecture Notes in Computer Science*), p. 741–755. Springer, 2009.

[12] Baecher (P.), Koetter (M.), Dornseif (M.) et Freiling (F.), « The nepenthes platform : An efficient approach to collect malware », dans *In Proceedings of the 9 th International Symposium on Recent Advances in Intrusion Detection (RAID)*, p. 165–184. Springer, 2006.

[13] Nicomette (V.), Kaâniche (M.), Alata (E.) et Herrb (M.), « Une analyse empirique du comportement d'attaquants — expérimentations et résultats », *RSTI-TSI*, vol. 29, n° 6/2010, 2010, p. 691–720.

[14] Ramsbrock (D.), Berthier (R.) et Cukier (M.), « Profiling attacker behavior following ssh compromises », *Dependable Systems and Networks, International Conference on*, 2007, p. 119–124.

[15] Lee (M.), Pincombe (B.) et Welsh (M.), « An empirical evaluation of models of text document similarity », dans *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, p. 1254–1259, 2005.