HAL
open science

Observation, caractérisation et modélisation de processus d'attaques sur Internet

Eric Alata

To cite this version:

Eric Alata. Observation, caractérisation et modélisation de processus d'attaques sur Internet. Réseaux et télécommunications [cs.NI]. INSA de Toulouse, 2007. Français. NNT : . tel-00280126

HAL Id: tel-00280126

https://theses.hal.science/tel-00280126

Submitted on 16 May 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE TOULOUSE

délivré par **l'Institut National des Sciences Appliquées de Toulouse**

Ecole Doctorale : Systèmes
Discipline : Systèmes Informatiques

présentée et soutenue

par

Eric ALATA

le 7 décembre 2007

Observation, caractérisation et modélisation de processus d'attaques sur Internet

Directeurs de thèse :

Mohamed Kaâniche
Vincent Nicomette

JURY

M. Yves Deswarte, Président
M. Ludovic Mé
M. Pierre Paradinas
M. Cédric Blancher
M. Marc Dacier

A mon petit frère,
Stéphan

A mes parents,
mes sœurs Annaïck et Pasquale
et ma Cathy

Remerciements

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes (Laas) du Centre National de la Recherche Scientifique (Cnrs). Je remercie Messieurs Malik Ghallab et Raja Chatila, qui ont assuré la direction du Laas-Cnrs depuis mon entrée, pour m'avoir accueilli au sein de ce laboratoire. Je remercie également Monsieur Jean Arlat, Directeur de Recherche CNRS, responsable du groupe de recherche Tolérance aux fautes et Sûreté de Fonctionnement informatique (Tsf), pour m'avoir permis de réaliser ces travaux dans ce groupe.

J'exprime ma très sincère reconnaissance à Messieurs Mohamed Kaâniche et Vincent Nicomette, respectivement Chargé de Recherche Cnrs et Maître de Conférence, pour m'avoir encadré, soutenu, et encouragé tout au long de cette thèse. Je les remercie pour leurs conseils, leur soutien et leur disponibilité. Je tiens également à souligner leurs compétences scientifiques dont j'ai beaucoup tiré parti. Je les remercie tous deux pour leur soutien constant et la confiance qu'ils m'ont toujours témoignée. Leurs lectures attentives des différentes versions du manuscrit et leur aide dans les moments de doute ont fortement contribué au bon déroulement des travaux présentés dans ce mémoire. Au delà de leur encadrement exemplaire, ils ont su créer et animer autour d'eux une équipe formidable avec laquelle j'ai pris un grand plaisir à travailler. Qu'ils trouvent ici un témoignage de mon estime et de ma reconnaissance.

J'exprime ma profonde gratitude à Monsieur Yves Deswarte, Directeur de Recherche Cnrs, pour l'honneur qu'il me fait en présidant mon jury de thèse, ainsi qu'à :

- Pierre Paradinas, Professeur au Cnam à Paris,
- Ludovic Mé, Professeur à Supélec à Rennes,
- Cédric Blancher, Ingénieur-chercheur à Eads France à Suresnes,
- Marc Dacier, Professeur à Eurécom à Sophia Antipolis,
- Mohamed Kaâniche, Chargé de Recherche au Laas-Cnrs à Toulouse,
- Vincent Nicomette, Maître de Conférences à l'Insa à Toulouse.

pour l'honneur qu'ils me font en participant à mon jury.

Je remercie particulièrement Messieurs Pierre Paradinas et Ludovic Mé qui ont accepté la charge d'être rapporteurs.

Mes remerciements vont naturellement à l'ensemble des membres du groupe Tsf avec lesquels j'ai pris beaucoup de plaisir à partager ces trois années de travail. Je n'oublie pas l'ensemble des membres des services techniques, administratifs et logistiques du Laas-Cnrs, qui par leur efficacité et disponibilité, m'ont permis de travailler dans

de très bonnes conditions. Je tiens à remercier en particulier Matthieu Herrb du Service Informatique et Instrumentation, pour son soutien et assistance à la mise en place des expérimentations menées dans le cadre de mes travaux. Enfin, je remercie tous ceux qui de près ou de loin, m'ont aidé ou soutenu dans mon travail. Je pense plus particulièrement à ma famille, mes amis et Cathy.

Table des matières

Introduction **1**

1 Cadre des travaux et état de l'art **5**
1.1 Introduction 5
1.1.1 Historique et usage d'Internet 5
1.1.2 Naissance de la piraterie 6
1.1.3 Motivation des pirates informatiques 6
1.1.4 Approche à la compréhension du comportement des pirates . . 8
1.2 Présentation des menaces sur Internet - terminologie 8
1.2.1 La sûreté de fonctionnement 8
1.2.2 Les malveillances 10
1.3 Moyens pour faire face aux malveillances 10
1.3.1 Prévention des fautes 11
1.3.2 Tolérance aux fautes 12
1.3.3 Elimination des fautes 14
1.3.4 Prévision des fautes 15
1.3.4.1 L'évaluation ordinale basée sur des critères 15
1.3.4.2 L'évaluation quantitative basée sur des modèles . . . 16
1.3.4.3 L'évaluation expérimentale 17
1.4 Sondes et pots de miel 18
1.4.1 Les sondes 18
1.4.2 Les pots de miel 18
1.4.2.1 Historique des pots de miel 19
1.4.2.2 Les pots de miel basse interaction 19
1.4.2.3 Les pots de miel haute interaction 20
1.4.2.4 Les pots de miel intermédiaires 21
1.4.3 Les projets basés sur les sondes et les pots de miel 21
1.4.3.1 Les projets Caida et Internet Motion Sensor . 21
1.4.3.2 Le projet Dshield 22
1.4.3.3 Le projet Leurre.com 22
1.5 Conclusion : orientation et contributions de la thèse 23

2 Caractérisation des processus d'attaques à partir des pots de miel basse interaction **25**
2.1 Architecture des pots de miel et base de données 26
2.1.1 Architecture 26

2.1.2 La base de données 26
2.2 Vue globale et analyses préliminaires des données 28
2.2.1 Présentation et analyse préliminaire 28
2.2.2 Problème lié aux périodes de silence suspectes 32
2.2.3 Discussion 33
2.3 Méthodologie d'analyse 33
2.3.1 Notations et définitions 33
2.3.2 Méthodologie 35
2.4 Prétraitement des données 36
2.4.1 Méthodes d'identification des valeurs aberrantes 36
2.4.2 Identification des périodes de silence suspectes 38
2.4.3 Sélection des données 43
2.5 Modélisation des intervalles entre attaques 47
2.5.1 Estimation des paramètres : l'algorithme EM 48
2.5.2 Validation des modèles : tests statistiques 50
2.5.2.1 Test du χ^2 50
2.5.2.2 Test de Kolmogorov-Smirnov 50
2.5.3 Application aux processus d'attaque observés 51
2.5.4 Densités de probabilité empiriques 51
2.5.5 Modélisation 52
2.6 Analyse des corrélations 55
2.6.1 Evolution dans le temps du nombre de sessions 56
2.6.2 Régression linéaire et coefficient de corrélation 56
2.6.3 Modèle de régression 58
2.6.4 Corrélation entre les environnements 58
2.6.5 Corrélation en fonction de l'origine géographique 58
2.7 Propagation des attaques 61
2.7.1 Principe de la propagation des attaques 61
2.7.2 Modèle de propagation 61
2.7.3 Illustration 63
2.8 Conclusion 64

3 Développement d'un pot de miel haute interaction 67
3.1 Introduction 67
3.2 Caractéristiques des pots de miel haute interaction 67
3.2.1 La transparence 67
3.2.2 L'observabilité 68
3.2.3 La flexibilité 68
3.2.4 La nature 68
3.3 Implémentations de pots de miel haute interaction 69
3.3.1 Un pot de miel avec VMware 69
3.3.2 Uml comme pot de miel 70
3.3.3 Sebek 70
3.3.4 Uberlogger 71
3.4 Architecture du pot de miel haute interaction 72
3.4.1 Les objectifs et les données à collecter 72
3.4.2 L'observation des activités des attaquants 74

3.4.2.1 Remarques préliminaires 74
3.4.2.2 Fonctionnement de `ssh` 74
3.4.2.3 Moyens d'observation des connexions `ssh` 75
3.4.3 La redirection des rebonds 77
3.4.4 Architecture générale 79
3.5 Conception et implémentation 79
3.5.1 La modification du noyau des systèmes d'exploitation 80
3.5.2 Archivage des données collectées 81
3.5.3 Récupération des données archivées 82
3.5.4 Vue globale de la collecte des données 83
3.5.5 Le mécanisme de redirection des connexions 84
3.6 Déploiement 86
3.7 Conclusion 88

4 Caractérisation des attaques observées sur le pot de miel haute interaction 89
4.1 Aperçu de l'activité des connexions 89
4.1.1 Analyses préliminaires 90
4.1.2 Origine des connexions 92
4.1.3 Discussion et méthodologie 93
4.2 Construction de l'ensemble des sessions 93
4.2.1 Fenêtre glissante 94
4.2.2 Définition formelle d'une session 97
4.2.3 Choix de la valeur du seuil 98
4.3 Identification des classes de comportement 100
4.4 Le processus d'attaque 102
4.5 Etude des attaques par dictionnaire 103
4.5.1 Présentation des attaques par dictionnaire observées 103
4.5.2 Distance inter-textuelle 106
4.5.3 Définition de la distance entre deux vocabulaires 107
4.5.4 Partitionnement des données 110
4.5.5 Classement des vocabulaires 112
4.5.6 Identification des cœurs de dictionnaire 113
4.6 Etude des intrusions 116
4.6.1 Identification des différentes communautés 116
4.6.2 Nature des intrus : êtres humains ou outils automatiques . . . 119
4.6.3 Les activités des intrus 121
4.6.4 Compétences des intrus 124
4.6.5 Enseignements apportés par l'emploi du mécanisme de redirection 125
4.7 Conclusion 126

Conclusion générale 129

Bibliographie 135

Introduction

Internet se développe très vite. Selon l'*Internet World Stats*[IWS], le nombre d'internautes recensés dans le monde serait de l'ordre de $1,1$ milliards. Ce réseau séduit par les services rendus accessibles, tels que le courrier électronique, les comptes bancaires, les offres d'emploi, l'achat de billets d'avion, etc. Cette attirance, en partie motivée par la cassure des barrières linguistiques, culturelles et sociales, fait naître de nouvelles communautés. Que ce soit les communautés des joueurs, des artistes, ou encore des professionnels, toutes ces communautés profitent de ce réseau et sont tributaires de son bon fonctionnement, au point que des lois permettent d'en dicter les règles d'utilisation. Pourtant, une communauté d'internautes particulière vient entacher le tableau : celle des pirates informatiques. Ces personnes utilisent l'outil informatique comme une arme pour commettre des délits. Ils profitent des vulnérabilités des logiciels et des matériels pour pénétrer les systèmes critiques, abuser des courriers éléctroniques, s'emparer de données confidentielles et inonder le réseau de logiciels malveillants. De plus, des instructions sur la manière d'utiliser les outils malveillants sont diffusées sur Internet, dans le but d'initier de nouveaux pirates. Le nombre d'incidents reportés et liés à ces délits ne cesse de croître. Ces problèmes préoccupent les communautés liées à la sécurité informatique et les rendent très actives.

Les experts en sécurité informatique étudient ces menaces pour mettre en place des contre-mesures. Plusieurs techniques peuvent être appliquées. Tout d'abord, les logiciels peuvent être conçus de manière à prevenir les vulnérabilités, par exemple en appliquant de bonnes pratiques d'ingénierie. Ensuite, une analyse du logiciel après développement par des techniques rigoureuses d'élimination des vulnérabilités permet d'établir des correctifs. Puis, des techniques de tolérance permettent d'assurer qu'un système remplisse sa fonction en dépit de la présence de vulnérabilités résiduelles. Toutes ces techniques tentent d'être des plus réactives pour prendre en compte les nouvelles vulnérabilités, découvertes par les pirates informatiques. Une course s'est donc lancée entre les pirates qui découvrent continuellement de nouvelles vulnérabilités et les experts en sécurité qui mettent en place de nouvelles techniques pour y faire face. Aujourd'hui, il nous semble nécessaire de pénétrer un peu ce milieu des pirates de façon à mieux connaître leur façon d'agir et ainsi mieux anticiper les nouvelles attaques.

Dans ce contexte, une bonne approche consiste à observer et étudier le comportement des pirates et exploiter cette connaissance pour mieux comprendre la menace et constituer une base de connaissances. Les analyses basées sur cette dernière devraient nous permettre de savoir *quand* et *comment* se protéger des pirates. Ce savoir pourrait alors être exploité pour guider le développement des techniques de la sécurité informatique, afin de définir des mécanismes appropriés pour faire face aux malveillances.

Les pots de miel constituent des outils adaptés à l'observation du comportement des attaquants. Un pot de miel est un système informatique volontairement vulnérable à une ou plusieurs vulnérabilités et visant à attirer les attaquants afin d'étudier leur comportement. Ils se déclinent en plusieurs catégories, basse ou haute interaction, en fonction du niveau d'interaction proposé aux pirates informatiques. Ils permettent de collecter des données issues de processus réels, primordiales pour établir des hypothèses réalistes sur le comportement des attaquants. Les travaux présentés dans ce manuscrit se basent sur les pots de miel afin d'apporter des informations importantes sur le comportement des pirates informatiques, voire pour étayer des intuitions ou briser des idées reçues sur ces comportements.

Il existe actuellement plusieurs plates-formes et environnements de collecte de données sur les attaques sur Internet. Ils utilisent principalement des pots de miel basse interaction. Plusieurs travaux existent également sur l'exploitation des données issues des pots de miel basse interaction pour caractériser les activités malveillantes selon différents critères (services et vulnérabilités ciblés, origine des attaques, analyse comparative des empreintes observées sur différents sites, etc.). Des avancées significatives ont été obtenues dans ce domaine et se sont concrétisées par le développement de méthodologies permettant d'extraire de façon semi-automatique des informations pertinentes sur les activités d'attaques observées. Ces études ont aussi montré que les données collectées par les pots de miel basse interaction peuvent fournir des informations très intéressantes à condition d'utiliser des méthodologies d'analyse rigoureuses. Cependant, à notre connaissance il n'existe pas encore de travaux ayant permis d'élaborer des modèles probabilistes et statistiques caractérisant les processus d'attaques observés. De tels modèles sont utiles pour générer des traces de trafic malveillants utiles pour des tests de validation de systèmes et mécanismes de sécurité, et également pour fournir les distributions et paramètres nécessaires pour les travaux visant à effectuer des évaluations prévisionnelles de la sécurité. Par ailleurs, la plupart des travaux existants dans ce domaine sont basés sur des pots de miel basse interaction qui ne sont pas adaptés pour collecter des informations sur les activités des attaquants qui ont réussi à s'introduire au sein d'un système cible. Les pots de miel haute interaction sont plus à même de fournir de telles informations mais le risque est plus important. Les résultats présentés dans ce manuscrit visent à apporter des réponses sur ces deux volets et sont tout à fait complémentaires aux avancées obtenues dans ce domaine.

Nos travaux ont été menés dans le cadre du projet Cadho « Collecte et Analyse de Données d'attaques basées sur des Honeypots », en collaboration avec l'Institut Eurécom et le Cert-Renater. Ce projet d'une durée de trois ans a débuté officiellement en septembre 2004. Les trois objectifs principaux sont : 1) de proposer une plate-forme de pots de miel basse interaction et la déployer sur Internet à divers endroits du globe, sur des sites académiques et industriels, afin de collecter des données permettant de faire des analyses comparatives des attaques et de leur propagation 2) de développer des méthodologies permettant d'extraire des informations pertinentes et de modéliser les activités malveillantes observées sur les différentes plateformes et de valider ainsi l'utilité de ces plates-formes 3) d'explorer la faisabilité d'un déploiement basé sur des pots de miel haute interaction permettant de collecter des informations sur les stratégies des attaquants une fois introduits au cœur d'un système cible.

Ce manuscrit est structuré en quatre chapitres. Le premier chapitre s'intéresse

aux différentes approches de la sécurité informatique depuis les approches classiques jusqu'à l'utilisation des pots de miel. Plus précisément, nous commencons par brosser un historique du phénomène de piraterie sur Internet. Ensuite, nous abordons les différentes techniques permettant d'y remédier. Pour finir, nous présentons les pots de miel et les projets associés et nous résumons les principales orientations de nos travaux.

Le deuxième chapitre concerne les analyses réalisées sur la base des pots de miel basse interaction. Dans un premier temps, nous présentons la méthodologie employée pour mener les analyses. Ensuite, nous étudions en détail les données collectées par ces dispositifs afin d'identifier une période minimisant l'impact des périodes d'indisponibilité pouvant affecter les plates-formes de collecte de données. Sur la base des données relatives à cette période, nous étudions trois aspects différents : la modélisation des intervalles entre attaques, la propagation des attaques à travers les différents pots de miel et les corrélations entre les activités observées sur les différents pots de miel.

Le troisième chapitre présente l'architecture et l'implémentation du pot de miel haute interaction que nous avons développée. Son objectif est de collecter des données concernant le comportement des pirates informatiques, une fois introduits au cœur du système. Pour leur permettre de s'y introduire, nous leur proposons une vulnérabilité des plus anciennes : les mots de passe simples à deviner. Nous commencons par examiner les différentes implémentations déjà disponibles. Nous confrontons ensuite ces implémentations à nos besoins. Nous proposons une architecture dont nous détaillons ensuite l'implémentation.

Le quatrième chapitre s'intéresse à l'analyse des données collectées sur le pot de miel haute interaction présenté au troisième chapitre. Nous commencons par présenter la méthodologie d'analyse. Ensuite, nous identifions deux comportements différents. Le premier concerne le processus suivi par les pirates informatiques afin de deviner les mots de passe du pot de miel. Le second concerne les activités réalisées au cœur du système, après utilisation des mots de passe découverts. Chacun de ces deux comportements sont ensuite analysés en détail.

Pour finir, la conclusion générale synthétise les principaux résultats obtenus dans le cadre de nos travaux et présente quelques perspectives.

Chapitre 1

Cadre des travaux et état de l'art

1.1 Introduction

Dans ce chapitre, nous présentons l'évolution d'Internet, de sa création aux différentes communautés utilisant ce réseau. Nous présentons les communautés de pirates informatiques qui constituent le sujet de nos travaux. Ensuite, nous abordons les différents moyens pour faire face à ces pirates informatiques. Parmi ces moyens, nous nous sommes focalisés sur l'évaluation expérimentale dans le cadre de la sécurité informatique, qui a pour objectif de collecter des données sur des processus d'attaque sur Internet et de fournir des informations pertinentes sur les activités malveillantes sur Internet. Nous mettons l'accent sur les travaux focalisant sur la collecte de données utilisant en particulier les pots de miel. Nous présentons brievement différents types de pots de miel et de projets travaillant sur ce thème. Nous résumons enfin les principales orientations de nos travaux de thèse.

1.1.1 Historique et usage d'Internet

Dans une atmosphère de guerre froide, le premier satellite artificiel est lancé en 1957 par l'URSS : Spoutnik. Cet exploit marqua l'importance des progrès technologiques. Il fit aussi prendre conscience que plus personne n'est à l'abri d'une attaque par missile nucléaire. Le président des ETATS-UNIS, Dwight David Eisenhower, créa l'Agence des Projets de Recherche Avancée (ARPA) afin de revenir dans une course technologique dominée par l'URSS. Une des premières missions de l'ARPA fut de concevoir un réseau capable, entre autres, de résister à des attaques nucléaires[LIV].

La mise en place d'un tel réseau reposa sur les travaux de Paul Baran[Bar64]. Il étudia trois organisations différentes : centralisée, décentralisée et distribuée. Ses résultats montrent qu'une organisation décentralisée permet à deux machines disponibles du réseau de communiquer même si d'autres machines ne sont pas disponibles. La transmission des données entre les machines du réseau s'appuie sur les travaux de Leonard Kleinrock[Kle61]. La théorie présentée dans cette thèse propose un système qui découpe les données en paquets lors de leur transmission. En 1969, un réseau décentralisé est mis en place entre quatre universités américaines : l'Université de Californie à Los Angeles (UCLA), l'Institut de recherche de Stanford (SRI), l'Université de Californie à Santa Barbara (UCSB) et l'Université de l'Utah. Ce réseau se nomme Arpanet. Il fut appelé Internet à l'occasion de la publication de la première RFC

(standards des protocoles Internet).

A l'origine, Internet était dédié à un usage scientifique. La naissance du World Wide Web[BL89] et le développement des navigateurs Web ont favorisé l'ouverture de ce réseau au grand public. Au début de l'année 2007, plus de 430 millions d'adresses Internet utilisées ont été recensées[ISC], trois fois plus qu'au début de l'année 2002. Aujourd'hui, Internet est fréquenté par des communautés diverses. Les entreprises voient en ce réseau un moyen de donner une plus grande visibilité à leur devanture. Les enfants peuvent exploiter de nouvelles ressources ludiques. Quant aux pirates informatiques, Internet constitue pour eux un bon moyen pour arriver à leurs fins : vol de données, exploits, etc.

1.1.2 Naissance de la piraterie

Le programme Elk Cloner est le premier à s'être répandu hors d'un laboratoire, en 1982, avant même la formalisation des virus[Ove07]. Son créateur, un étudiant de 15 ans, lui donna la capacité d'infecter un ordinateur au démarrage, depuis une disquette infectée. En 1983, Fred Cohen développa, à des fins de recherche, un des premiers programmes possédant la capacité d'infecter d'autres programmes et de se reproduire[Coh87]. Leonard Adleman baptisa ce programme *virus*[Adl90].

En 1988, le premier ver ayant une conception adaptée pour Internet fit son apparition. Ce ver nommé Morris, du nom de son concepteur, causa de gros dégats en raison de la forte connectivité des machines sur Internet. Sa vitesse de propagation importante suscita l'intérêt des pirates informatiques. Depuis, elle a nettement été augmentée. Des travaux ont montré qu'il est possible, en théorie, d'infecter plus de 300 000 machines en moins de 15 minutes[SPW02]. Cette vitesse, bien que théorique, a été approchée par le ver Code Red, lancé à l'encontre des serveurs Web de Microsoft en 2001. Il a infecté plus de 350 000 machines en moins de 14 heures[pro].

Les virus n'ont cessé d'évoluer[CR04, Fil04]. Ils profitent de nouveaux vecteurs de propagation. En 2005, le premier virus sur téléphones mobiles a été détecté : Commwarrior. Il utilise la technologie des Mms pour se répandre. Ces logiques malignes ne sont pas les seules armes des pirates informatiques. Les attaques par déni de service sont aussi très répandues. Elles visent à inonder un serveur de requêtes afin de le rendre indisponible. De nombreux serveurs sur Internet en sont victimes : Google, Microsoft, le site du Pentagone, etc. Le nombre d'incidents reporté n'a cessé d'augmenter[CER07], tout comme le nombre de vulnérabilités identifiées. Les pirates informatiques sont omniprésents sur Internet et plus organisés. Ils se regroupent en communautés. L'une des communautés les plus célèbres est le « Cult of the death Cow ». Ce groupe allemand de pirates est à l'origine de plusieurs outils destinés aux pirates et aux administrateurs de systèmes informatiques. Le piratage est un phénomène de société très complexe.

1.1.3 Motivation des pirates informatiques

Internet tient une place de plus en plus importante dans le commerce, les loisirs et la vie politique. Il constitue un lieu privilégié pour les pirates informatiques. Le pourquoi de leurs agissements trouve sa réponse au niveau de leurs motivations. Elles peuvent être de plusieurs natures[Des03]. Elles révèlent soit des ambitions économiques, soit des positions politiques, soit une curiosité simplement ludique.

Pour les trafics de drogue, d'armes et d'argent, le recrutement d'intermédiaires brouille les pistes. Certains d'entre eux, les mules, jouent le rôle de passeur, parfois à leur insu. Ils se rendent alors coupables de complicité de fraude. Dernièrement, ce principe a été adapté au monde informatique[Clu07]. Les activités de blanchiment, de vol et de détournement d'argent sont à l'origine d'une recrudescence de *spam*. Le principe est simple. Un fraudeur rédige un message électronique en masquant ses intentions. Le paravant d'une action humanitaire est souvent utilisé. Le message est envoyé à plusieurs internautes. Chacun se voit sollicité pour devenir collaborateur d'une soi-disant transaction. Le contrat tient en quelques lignes : accepter de recevoir une somme d'argent sur son compte, la transmettre sur un autre compte tout en empochant une petite somme. Le dépôt de cette somme sur le compte destinataire devient légitime. Si ce contrat est alléchant pour l'internaute, il le place dans une situation délicate : complicité de blanchiment d'argent.

Un autre scénario mettant en évidence les ambitions économiques des pirates informatiques concerne les *bookmaker*. Ces personnes, morales ou physiques, permettent de parier sur divers évènements. Leur activité est intense. La moindre seconde d'indisponibilité de leur service peut entraîner de lourdes pertes. Ainsi, des pirates informatiques exploitent cette faille pour rançonner les *bookmaker*[Ber03], l'une de leurs cibles privilégiées. Ils leur envoient des messages électroniques pour les menacer d'attaques de grandes envergures visant à rendre indisponible leur service. Leur demande correspond au versement d'une somme ridicule face aux dommages occasionnés par les indisponibilités. Pour le *bookmaker*, le pari est risqué : céder au chantage et verser une somme ridicule ou ignorer les menaces et risquer la disponibilité de son service. Ces activités sont fréquentes. L'une des dernières victimes en date est le site `winamax.com`[Ban07], appartenant à une société anglaise de pari en ligne.

Les pirates informatiques animés de motivations politiques sont nommés *hacktivistes*. Ils mettent à profit leurs savoirs sur les techniques informatiques pour défendre leurs idées. Parfois, les motifs de leurs actes transparaissent nettement dans les conséquences. Par exemple, le piratage des sites de personnalités politiques est souvent porteur d'un message de mécontentement affiché en clair. D'autres fois, le lien entre le piratage et les faits politiques est plus subtil. Prenons, à titre d'exemple, le cas des attaques qui ont ciblé le réseau estonien survenues en avril 2007. Pendant plusieurs jours, le réseau estonien a du faire face à des attaques de grande envergure. Il a été submergé par un nombre important de connexions. Le résultat a été l'indisponibilité du réseau estonien, entraînant des conséquences fâcheuses pour les utilisateurs quotidiens[Jac07]. Selon l'Asymmetric Threats Contingency Alliance (Atca), une association composée d'experts internationaux, la Russie serait mêlée à ces attaques. Elle aurait loué les services de *botnets*, ensemble de machines exploitées conjointement de manière malveillante, pour donner à ces attaques une dimension imparable. Quelques jours auparavant, l'Estonie envisageait le déplacement du monument érigé à la mémoire des soldats russes morts pendant la seconde guerre mondiale, sur le sol estonien.

Une autre motivation des plus importantes est d'ordre ludique. Pour le plaisir de repousser leurs limites ou simplement pour assouvir leurs curiosités, des pirates informatiques mettent en pratique leurs connaissances en informatique pour pirater sur Internet. Parfois, leur identité est retrouvée par les autorités. Souvent, ils utilisent des recettes de cuisine obtenues sur Internet. Dans tous les cas, ces méfaits occasionnent

des gênes incontestables. Pour illustrer ces propos, citons quelques exemples issus de l'actualité. Un individu de 28 ans a été arrêté en juin 2007 pour avoir lancé un virus ayant infecté 115000 téléphones portables[Pir07]. Un internaute a été interpellé pour avoir penétré les sites du pentagone et de la NASA, à la recherche d'informations sur les ovnis[Cab07].

1.1.4 Approche à la compréhension du comportement des pirates

Pour faire face aux dégats qu'engendre le piratage, des mécanismes et outils de protection ont été développés en tant que contre-mesure, par exemple des pare-feux, des outils de detection d'intrusion, etc. Malgré ces efforts, le nombre d'incidents reporté ne cesse de croître[CER07]. Une bonne compréhension des menaces et du comportement des attaquants est nécessaire pour améliorer l'efficacité de mécanismes de protection vis-à-vis des malveillances. Aussi, une étude directe sur les comportements des communautés de pirates nous permettrait de mieux penser ces outils de contre-mesure.

Dans [Hum06], l'auteur privilégie une enquête de terrain. En abordant le problème de la cybercriminalité du côté des sciences humaines, il tente d'enrichir les connaissances sur les pirates informatiques. L'un des moyens mis en œuvre est un questionnaire en ligne[Hum] dont le but est de déterminer une représentation sociale du cybercriminel. Répondre à un questionnaire est un acte volontaire. Certaines communautés de pirates informatiques peuvent s'y refuser. Pour les mêmes motivations que précédemment, nous allons aborder le problème de la cybercriminalité du côté technique. En collectant des informations sur les pirates informatiques pendant leur actions et à leur insu, nous allons enrichir nos connaissances. Notamment, nous cherchons à observer leur comportement pour mieux nous protéger.

Le thème des travaux présentés dans ce manuscrit est l'observation, la caractérisation et la modélisation du comportement des attaquants. Dans la suite, nous commencons par présenter des éléments de terminologie pour mieux situer nos travaux par rapport au domaine général de la sûreté de fonctionnement.

1.2 Présentation des menaces sur Internet - terminologie

Précédemment, nous avons utilisé la notion de vulnérabilité et d'attaque. Elles s'inscrivent pleinement dans le domaine de la sûreté de fonctionnement. Nous allons les définir dans cette section. Mais, avant tout, il convient d'introduire une partie du vocabulaire, en considérant les concepts introduits dans le « Guide de la Sûreté de Fonctionnement »[L+96] et mis à jour dans [ALRL04]. Plus précisément, nous commencons par définir les notions de sûreté de fonctionnement, faute, erreur et défaillance pour ainsi aborder avec plus d'aisance la définition d'une vulnérabilité et situer nos travaux par rapport au thème et aux moyens classiques de la sûreté de fonctionnement en général et de la sécurité informatique en particulier.

1.2.1 La sûreté de fonctionnement

La sûreté de fonctionnement d'un système informatique est la propriété qui permet à ses utilisateurs de placer une confiance justifiée dans le service qu'il leur délivre. Le service délivré par un système est son comportement tel que perçu par ses utilisateurs.

Ces derniers peuvent être humains ou physiques. La sûreté de fonctionnement informatique s'articule autour de trois principaux axes : les *attributs* qui la caractérisent, les *entraves* qui empêchent sa réalisation et les *moyens* de l'atteindre (cf. figure 1.1).

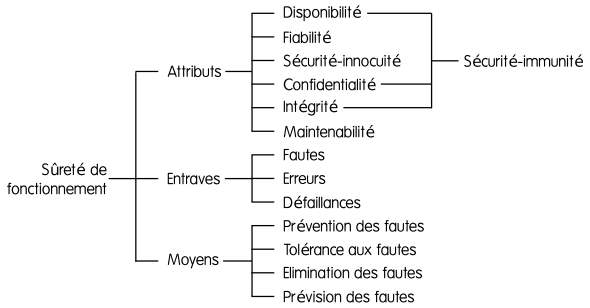


FIG. 1.1 – La sûreté de fonctionnement

La sûreté de fonctionnement d'un système peut être perçue selon différentes propriétés. Ces propriétés se nomment les *attributs* de sûreté de fonctionnement. Les attributs à considérer dépendent des applications auxquelles le système est destiné. Les six attributs sont les suivants :

- *la disponibilité* : aptitude du système à être prêt à l'utilisation,
- *la fiabilité* : continuité du service,
- *la sécurité-innocuité* : absence de conséquences catastrophiques pour l'environnement,
- *la confidentialité* : absence de divulgations non-autorisées de l'information,
- *l'intégrité* : absence d'altérations inappropriées de l'information,
- *la maintenabilité* : aptitude aux réparations et aux évolutions.

Les attributs précédents peuvent être mis à mal par des entraves. Une entrave à la sûreté de fonctionnement est une circonstance indésirable mais non inattendue. Elle est la cause ou le résultat de la non-sûreté de fonctionnement. Nous en distinguons trois sortes :

- *la défaillance* : survient lorsque le service délivré dévie de la fonction du système,
- *l'erreur* : partie de l'état de système susceptible d'entraîner une défaillance,
- *la faute* : cause adjugée ou supposée de l'erreur.

Ces fautes s'enchaînent pour former une chaîne fondamentale en sûreté de fonctionnement : faute → erreur → dafaillance → faute etc. Pour minimiser l'impact de ces entraves sur les attributs retenus d'un système, nous disposons de moyens. Ce sont des méthodes et des techniques permettant de conforter les utilisateurs dans le bon accomplissement de la fonction du système. Elles peuvent être utilisées conjointement pour le développement d'un système sûr de fonctionnement. Elles sont classées en quatre moyens :

- *la prévention* : empêche l'occurrence ou l'introduction de fautes,
- *la tolérance* : fournit un service qui remplit la fonction du système en dépit des fautes,
- *l'élimination* : réduit la présence (nombre, sévérité) des fautes,
- *la prévision* : estime la présence, la création et les conséquences des fautes.

Dans la suite de ce document, nous nous intéressons à l'association de la confidentialité, de l'intégrité et de la disponibilité, nommé *sécurité-immunité*[Lap04][ALRL04].

Plus précisément, nous traitons les fautes malveillantes.

1.2.2 Les malveillances

Les concepts de la sûreté de fonctionnement présentés dans le paragraphe précédent ont été conçus pour être génériques et applicables à différents contextes. Une adaptation au contexte particulier de la sécurité informatique et aux malveillances a fait l'objet de travaux effectués dans le cadre du projet MAFTIA[MAF03].

Le projet MAFTIA a porté sur le développement de techniques de tolérance aux fautes accidentelles et aux malveillances pour des applications réparties à grande échelle sur Internet. En particulier, une terminologie des malveillances a été introduite dans le cadre de ce projet. Nous en reprenons quelques définitions.

Les fautes malveillantes se déclinent en deux classes principales : les logiques malignes et les intrusions. Les logiques malignes sont des parties du système conçues pour provoquer des dégâts (bombes logiques) ou pour faciliter des intrusions futures (vulnérabilités créées volontairement). Elles peuvent être introduites dès la création du système (par un concepteur malveillant), ou en phase opérationnelle (par l'installation d'un logiciel contenant un cheval de Troie ou par une intrusion). La définition d'une intrusion est étroitement liée aux notions d'attaque et de vulnérabilité[MAF03] :

- *une attaque* : faute d'interaction malveillante visant à violer une ou plusieurs propriétés de sécurité. C'est une faute externe créée avec l'intention de nuire, y compris les attaques lancées par des outils automatiques : vers, virus, zombies, etc.
- *une vulnérabilité* : faute accidentelle ou intentionnelle (avec ou sans volonté de nuire), dans la spécification des besoins, la spécification fonctionnelle, la conception ou la configuration du système, ou dans la façon selon laquelle il est utilisé. La vulnérabilité peut être exploitée pour créer une intrusion.
- *une intrusion* : faute malveillante interne d'origine externe, résultant d'une attaque qui a réussi à exploiter une vulnérabilité qui peut produire des erreurs pouvant provoquer une défaillance vis-à-vis de la sécurité, c'est-à-dire une violation de la politique de sécurité du système.

Les moyens de la sûreté de fonctionnement ne sont pas limités aux fautes accidentelles. Effectivement, les intrusions, les attaques et les vulnérabilités sont des fautes. Or, les fautes peuvent être tolérées, éliminées, prévues, etc. Ces moyens peuvent être appliqués aux fautes malveillantes, afin d'améliorer la sécurité-immunité d'un système. Chaque moyen peut être appliqué sur les différentes fautes malveillantes (attaque, vulnérabilité et intrusion). Pour chaque moyen, les techniques à employer dépendent de la conjugaison entre le moyen et la faute. Le tableau 1.2 présente ces techniques en fonction des moyens et des fautes[MAF03]. La section suivante décrit ces méthodes, pour chaque moyen.

1.3 Moyens pour faire face aux malveillances

Nous nous intéressons plus particulièrement à la sécurité-immunité des systèmes. Dans cette section, nous présentons les moyens à notre disposition pour satisfaire ces attributs de la sûreté de fonctionnement des systèmes. Les quatre moyens sont

	attaque humaine	*attaque technique*	*vulnérabilité*	*intrusion*
prévention	lois, pression sociale, service secret	pare-feu, authentification, autorisation	spécifications formelles et semi-formelles, méthodes rigoureuses de développement	≡ prévention et élimination des attaques et vulnérabilités
tolérance	≡ prévention des vulnérabilités, élimination et tolérance aux intrusions		≡ prévention des attaques, élimination et tolérance aux intrusions	détection et recouvrement d'erreur, alertes, masquage de fautes, détection d'intrusion
élimination	contre-mesures physiques, capture de l'attaquant	maintenance préventive et corrective	preuve formelle, model-checking, inspection, test, maintenance préventive et corrective	≡ élimination des attaques et des vulnérabilités
prévision	collecte d'informations sur les attaques réussies	analyse des agents malveillants latents	recensement des vulnérabilités et des difficultés d'exploitation et des conséquences potentielles	≡ prévision des vulnérabilités et des attaques

FIG. 1.2 – Méthodes disponibles pour la mise en œuvre des moyens de la sûreté de fonctionnement

présentés : la prévention des fautes, la tolérance aux fautes, l'élimination des fautes et la prévision des fautes.

1.3.1 Prévention des fautes

Rappelons l'objectif de la prévention des fautes : empêcher l'occurrence ou l'introduction de fautes. Ce moyen prend place au début du cycle de vie du système, au moment de penser le développement et le déploiement. Il englobe les bonnes pratiques d'ingénierie et l'utilisation de politiques de sécurité.

Parmi les bonnes pratiques d'ingénierie, la spécification formelle occupe une place importante. Elle repose sur des bases mathématiques pour décrire ce que doit faire le système et non pas comment il doit le faire. Une validation est réalisée pour s'assurer que la description coïncide effectivement avec les attentes. Cette approche formelle permet de lever les ambiguités qui peuvent persister dans une spécification en langage naturel. La description peut être réalisée avec la notation Z ou la méthode B, par exemple. Elle sera utilisée à son tour lors de l'étape de développement.

Idéalement, le développement du système doit être, dans la mesure du possible, une « simple » traduction en langage de programmation de la description de ce système. Concernant ce développement, des normes ont été mises en place pour des secteurs d'activité sensibles. Elles imposent des règles dans l'utilisation des langages de programmation. Le but est de prévenir les fautes fréquentes. Par exemple, dans la norme Misra-C[MISRA] pour l'automobile, l'arithmétique sur les pointeurs est proscrite. Cette pratique évite l'introduction de fautes par ces techniques avancées de programmation. Par la même occasion, leur utilisation malveillante est ainsi prévenue. Les techniques d'ingénierie rigoureuse visant à empêcher l'occurrence de fautes lors des phases de développement ont été initialement appliquées pour faire face aux fautes accidentelles, par exemple dans le logiciel. Elles restent généralement applicables dans le cas des malveillances. Dans ce domaine, un effort important a aussi été consacré au développement de politiques de sécurité.

L'ensemble des propriétés de sécurité que l'on désire assurer dans un système ainsi que la façon dont on va les assurer sont définies dans la politique de sécurité du système. « La politique de sécurité d'un système est l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique »[ITS91]. Elle doit identifier les objectifs de sécurité du système et les menaces auxquelles celui-ci devra faire face. Cette notion de politique de sécurité peut être raffinée en trois branches distinctes : les politiques de sécurité *physique*, *administrative* et *logique*. La première s'occupe de tout ce qui touche à la situation physique du système à protéger. Les procédures administratives traitent de tout ce qui ressort de la sécurité d'un point de vue organisationnel au sein de l'entreprise. La politique de sécurité logique est en charge de réaliser tous les contrôles d'accès logiques au contenu du système informatique. Elle doit spécifier *qui* a accès à *quoi* et dans *quelles circonstances*. La politique logique peut se décomposer en plusieurs phases. Chaque individu qui utilise un système sécurisé doit s'identifier et doit pouvoir prouver qu'il est bien la personne qu'il prétend être. Ces deux phases sont définies dans la *politique d'identification* et dans la *politique d'authentification*. Une politique d'authentification peut être mise en place sur la base d'un système à mot de passe. Un utilisateur ayant fourni un mot de passe valide est authentifié. Lorsqu'un utilisateur est authentifié, la *politique d'autorisation* doit spécifier quelles sont les opérations que cet utilisateur particulier peut réaliser dans le système. Pour plus d'informations sur ce sujet, une description récente plus détaillée des modèles et politiques de sécurité est présentée dans [CC06].

1.3.2 Tolérance aux fautes

La tolérance aux fautes, telle qu'elle est définie dans la section 1.2.1, correspond à un ensemble de moyens destinés à assurer qu'un système remplit sa fonction en dépit des fautes. Elle est obtenue grâce à la mise en œuvre de techniques de traitement d'erreurs et de fautes[LA90], en deux étapes. La première étape, la *détection d'erreur*, permet d'identifier un état erroné. La seconde étape, le *rétablissement*, permet de ramener le système dans un état de confiance. Du point de vue des fautes malveillantes, un système tolérant aux intrusions est un système capable de s'auto-diagnostiquer, se réparer et se reconfigurer tout en continuant à fournir un service acceptable aux utilisateurs légitimes pendant les attaques[DBF91].

Les méthodes de détection des intrusions visent à détecter des atteintes à la politique de sécurité d'un système. Deux approches existent : *l'approche comportementale* et *l'approche par scénario*. La première[DMD06] se base sur l'hypothèse qu'un comportement malveillant entraîne une activité inhabituelle sur le système. Elle cherche à répondre à la question : « le comportement actuel de l'utilisateur et/ou application est-il cohérent avec son comportement passé ? ». Ainsi, une atteinte à la politique de sécurité d'un système est détectée lorsqu'un comportement observé est non référencé dans cette base de comportements normaux. Une base d'attaques n'est pas nécessaire. Des activités malveillantes basées sur des attaques inconnues peuvent alors être détectées. Par contre, les comportements erratiques sont difficiles à évaluer. Une implémentation de cette approche est l'outil `eTrust`[ETR]. Quant à l'approche à base de scénario, elle utilise une base de signatures de scénarios anormaux. Elle détecte une atteinte à la politique de sécurité lorsqu'une séquence d'informations – liée à un comportement – possède une signature référencée dans la base de signatures de scénarios anormaux. Cette base doit être constamment mise à jour, sous peine de ne pas détecter les attaques non connues. Une implémentation répandue de cette approche est l'outil `snort`[Sno02]. De manière générale, les mécanismes de détection d'intrusions soulèvent un nombre important de fausses alertes et certaines violations de la politique de sécurité ne sont pas détectées. L'analyse de ces alertes est rendue difficile. Aussi, afin d'obtenir des outils plus efficaces, des travaux proposent d'utiliser différents outils de detection d'intrusion conjointement[MMM+01, CM02]. Un état de l'art récent sur les techniques de detection d'intrusion est présenté dans [DMD06] et une bibliographie imposante est présentée dans [MM01].

Un système tolérant aux intrusions est un système capable de s'auto-diagnostiquer, se réparer et se reconfigurer tout en continuant à fournir un service acceptable aux utilisateurs légitimes pendant les attaques[DBF91, DP06]. La tolérance aux intrusions peut être appliquée avec différentes techniques de sécurité. Parmi les techniques classiques de sécurité, nous pouvons citer le chiffrement, la réplication, et le brouillage des données. La technique de fragmentation, redondance et dissémination a pour but une approche globale de la tolérance aux fautes accidentelles et intentionnelles, pour le traitement, le stockage et la transmission d'informations confidentielles. Pour atteindre ce but, elle découpe l'information en fragments, duplique ces fragments et les disperse sur différents sites. Les conditions que doit remplir le pirate pour reconstituer la données sont plus nombreuses, rendant sa tâche nettement plus difficile. Concernant la redondance avec diversification, l'idée sous-jacente repose sur un constat simple : une attaque qui cible une vulnérabilité d'un système fonctionnant sur une architecture matérielle particulière a peu de chance de fonctionner sur un autre système fonctionnant sur une autre architecture matérielle. Le principe de cette technique est donc d'utiliser plusieurs sous-systèmes différents sur des architectures matérielles différentes et fournissant le même service, avec un mécanisme de vote majoritaire pour former le système globale. De la sorte, une attaque susceptible d'engendrer des dégats sur un des sous-systèmes sera inéfficace sur les autres. Le système global sera toujours disponible sous réserve, au moins, que la majorité des sous-systèmes soient aussi disponibles.

Une des premières architectures tolérantes aux intrusions a été développée dans le cadre du projet Delta-4 dans les années 1980, pour des serveurs de stockage de données, d'authentification et d'autorisation. Plus recemment, l'utilisation des principes de tolérance aux intrusions a été employée dans d'autres projets, par exemple

[VNC+06, BSC+07, VAC+]. Notamment, le projet DIT[VAC+] propose une architecture tolérante aux intrusions basée sur une diversification de serveurs Web et d'architectures matérielles. Les travaux réalisés dans le projet MAFTIA ont visé à développer des politiques d'autorisation plus efficaces en se basant sur des techniques de fragmentation, redondance et dissémination et les cartes à puces Java. Les politiques proposées dans ce projet sont tolérantes aux fautes accidentelles et aux intrusions.

1.3.3 Elimination des fautes

Malgré l'application de techniques de prévention de fautes et en raison de la difficulté de développer correctement les systèmes, certaines fautes peuvent persister. Ce sont les fautes de conception, par exemple l'absence de test d'un indice dans un tableau. Ces fautes peuvent être exploitées à des fins malveillantes. L'élimination des fautes vise à en réduire le nombre. Ce moyen s'appuie sur des analyses statiques (revues, inspections), des vérifications formelles ou semi-formelles, des techniques de test, etc.

La vérification formelle est une technique permettant de démontrer que les propriétés attendues du système sont vérifiées. Elle se base sur la description formelle du système et une liste des propriétés à vérifier. Une approche courante consiste à appliquer des outils de *model-checking* (SMV, NuSMV, ...) pour identifier des contre-exemples éventuels. De plus, certaines méthodes permettent l'analyse du code source du système en vue de détecter des fautes particulières, telles que les dépassements de tampon[AH07][HB03][LE01]. Toutefois, la technique de vérification formelle ne suffit pas à elle seule. Elle doit être suivie d'un diagnostic permettant de localiser la faute, pour aboutir à la création d'un correctif : un greffon. Certains travaux ont permis, à partir d'un greffon et du système à greffer, d'identifier la faute pour, éventuellement, une exploitation maligne[Fla05]. Une bonne pratique consiste donc à appliquer ce greffon dès sa disponibilité.

Le test est une technique largement utilisée pour révéler les fautes résiduelles. Il consiste à activer le système réel ou un prototype avec des entrées spécifiques caractérisant des activités normales ou anormales du système et de vérifier si les propriétés attendues sont satisfaites à partir des sorties observées. Différents types de test existent en fonction des critères considérés pour la sélection et la génération des entrées. En particulier, le test de robustesse basé sur l'injection de fautes permet de tester la réaction du système cible dans des conditions extrêmes et en présence d'entrées valides et invalides. Deux types distincts de domaines d'entrée sont distingués : les activités et les fautes. Le test à base d'activités permet de mettre sous pression le système afin d'observer son attitude à y faire face, comme réalisé dans [WTF99]. Le test à base de fautes permet de soumettre délibérement des fautes en entrée du système, telles que des changements de bit[ACK+03] ou des instances de classe incorrectes[CS04]. L'objectif est d'étudier les effets des fautes qui peuvent affecter le système évalué pendant son exécution.

Dans le domaine de la sécurité, outre les techniques classiques de test vis-à-vis des fautes de conception, des techniques spécifiques ont été développées pour l'analyse des malveillances. Nous citons par exemple les tests de pénétration par des experts en sécurité (appelés parfois *red team* ou *tiger team*) qui tentent de violer les objectifs de sécurité en contournant les mécanismes de protection. Les techniques d'élimination

de fautes incluent également des vérifications de configuration du système et des vulnérabilités présentes en analysant le système de fichier et en s'appuyant sur des bases de données de vulnérabilités connues. C'est l'approche employée, par exemple, dans des outils de recherche de vulnérabilités tels que `nessus`[Lau02], `esope`[ODK99] ou `cops`[DS90]. Nous pouvons aussi citer des techniques de vérification de l'intégrité des fichiers du système pour s'assurer qu'ils n'ont pas fait l'objet de modifications malveillantes.

1.3.4 Prévision des fautes

La prévision des fautes a pour objectif d'identifier les vulnérabilités, fautes, erreurs et modes de défaillances potentiels du système et d'analyser voire quantifier leur impact sur le comportement de ce système, vis-à-vis des propriétés attendues. Elle peut être conduite en s'appuyant sur différents types d'évaluations qui sont complémentaires : des évaluations 1) qualitatives ou ordinales, 2) quantitatives ou probabilistes à base de modèles ou 3) expérimentales. Dans la suite, nous présentons brièvement ces trois méthodes et leur utilisation pour l'analyse des malveillances et l'évaluation de leur impact sur la sécurité.

1.3.4.1 L'évaluation ordinale basée sur des critères

L'évaluation ordinale, ou qualitative, est destinée à identifier, classer et ordonner les défaillances ou les méthodes et techniques mises en œuvre pour les éviter, par un organisme indépendant. Elle se base par exemple sur des critères d'évaluation dont le but est de donner confiance à l'utilisateur dans le système. Les premiers travaux ont été initié par le conseil scientifique de l'armée américaine (Defense Science Board) afin étudier les différentes approches de protection des données classifiées dans les systèmes informatiques. Parmi leurs productions, le document *Trusted Computer Security Evaluation Criteria* »(Tcsec) datant de 1985, plus connu sous le nom de « livre orange », est devenue une norme du Département de la Défense américaine (DoD)[TCS85]. Ce document est la référence en matière d'évaluation de la sécurité des systèmes informatiques. Les critères offrent une classification à sept niveaux de sécurité (A1, B1, B2, B3, C1, C2), regroupés en quatre classes (A, B, C, D). Quatre familles de critères sont définies pour chaque niveau. Elles traitent respectivement de *la politique d'autorisation*, de *l'audit*, de *l'assurance* et de *la documentation*. L'évaluation d'un produit consiste à lui attribuer un des sept niveaux de sécurité. Cette attribution n'aboutit que si le produit répond à tous les critères du niveau en question. La politique d'autorisation employée ne permet pas de prendre en compte les contraintes liées aux pratiques commerciales. Cette limite a suscité la création de nouveaux critères dans d'autres pays, avec, par exemple, les « *Canadian Trusted Computer Product Evaluation Criteria* »[CTC93] (Ctcpec) pour le Canada et les « *Japanese Computer Security Evaluation Criteria* »[JCS92]. Nous citons ci-dessous l'exemple des « Information Technology Security Evaluation Criteria »(Itsec) adopté par la Communauté Europèenne et des « Critères Communs »(Cc). Les Itsec sont le résultat de l'harmonisation de travaux réalisés au sein de quatre pays européens : l'Allemagne, la France, les Pays-Bas et le Royaume-Uni[ITS91]. Les Itsec se différencient essentiellement du livre orange par la définition d'un certain nombre de classes de fonctionnalités

d'une part et un certain nombre de classes d'assurance d'autre part. Une classe de fonctionnalité décrit les mécanismes que doit mettre en œuvre un système pour être évalué à ce niveau de fonctionnalité. Une classe d'assurance permet, quant à elle, de décrire l'ensemble des preuves qu'un système doit apporter pour montrer qu'il implémente réellement les fonctionnalités qu'il prétend assurer. Les ITSEC utilisent le terme *cible d'évaluation* (*Target of Evaluation* ou TOE). Le contenu d'une cible d'évaluation comprend : une politique de sécurité, une spécification des fonctions requises dédiées à la sécurité, une définition des mécanismes de sécurité requis et le niveau d'évaluation visé. Les critères communs (CC) [CC999, CCS] sont nés de la tentative d'harmonisation des critères canadiens (CTCPEC), des critères européens (ITSEC) et des critères américains (TCSEC). Ils contiennent deux parties bien séparées comme dans les ITSEC : fonctionnalité et assurance. De même que dans les ITSEC également, les CC définissent une cible d'évaluation et une *cible de sécurité* (*Security Target*). Une cible d'évaluation désigne le système ou le produit à évaluer. La cible de sécurité contient les objectifs de sécurité d'une cible d'évaluation particulière. La cible de sécurité pour une cible d'évaluation représente la base d'entente entre développeurs et évaluateurs. Elle peut contenir les exigences d'un ou de plusieurs profils de protection (*Protection Profiles*) prédéfinis. Une des différences essentielles entre ITSEC et CC réside dans l'existence de ces profils, qui avaient auparavant été introduits dans les Critères Fédéraux[Tec92]. Un profil de protection définit un ensemble d'exigences de sécurité et d'objectifs, indépendants d'une quelconque implémentation, pour une catégorie de cible d'évaluation.

1.3.4.2 L'évaluation quantitative basée sur des modèles

Les méthodes d'évaluation ordinale de la sécurité basées sur des critères sont utiles pour guider le développement des systèmes. Cependant, elles ne sont pas adaptées pour estimer l'impact des fautes résiduelles, effectuer des analyses comparatives en considérant différentes variantes d'architectures ou différentes configurations du système, ou bien pour suivre l'évolution de la sécurité en opération. De telles analyses sont courantes quand on considère des fautes accidentelles. En effet, plusieurs méthodes basées sur des modélisations analytiques sont utilisées dans cette optique dans le domaine de l'évaluation de la fiabilité, la disponibilité ou la sécurité innocuité des systèmes. On peut citer par exemple des modèles basés sur les arbres de fautes, les diagrammes de fiabilité, les chaînes de Markov, etc. Dès les années 1990, des travaux ont été effectués dans l'objectif d'étendre ces techniques et de développer des mesures et des méthodes d'évaluation quantitatives adaptées à la sécurité-immunité et aux malveillances. Un tour d'horizon sur les travaux menés dans ce cadre et sur les technique d'évaluation quantitative de la fiabilité et de la sécurité à base de modèles est effectué dans [NST04]. Les travaux sur l'évaluation quantitative de la sécurité ont démarré en particulier dans le cadre du projet européen PDCS en collaboration avec l'Université de City (GB), Le LAAS-CNRS et l'Université de Chalmers (SE). Dans [BLOJ94], les auteurs ont étudié les similarités entre la fiabilité et la sécurité dans le but d'obtenir une mesure de la sécurité similaire aux mesures de fiabilité existantes. Notamment, ils ont défini ce que doit représenter une mesure de la sécurité : « la capacité d'un système à résister aux attaques ». Pour valider cette approche, des expérimentations ont été menées où des étudiants ont joué le rôle de pirates pour péné-

trer un système. L'approche présentée dans[BLOJ94] est de type « boîte noire ». Une approche de type « boîte blanche » basée sur de principes similaires a été développée dans [DKD93, Dac94]. En particulier, un modèle formel appelé « graphe des privilèges » a été défini pour décrire les vulnérabilités du système pouvant être exploitées par des attaquants pour mettre en défaut des objectifs de sécurité [DD94]. Ce graphe sert ensuite à générer des scénarios d'attaques en considérant des hypothèses sur la stratégie qu'un attaquant va adopter pour exploiter ces vulnérabilités. L'analyse de l'association de ces scénarios aux efforts permet d'obtenir des mesures quantitatives pour caractériser la sécurité. Un outil mettant en œuvre cette approche (Esope) a été aussi développé. Une expérimentation sur le réseau du LAAS a permis de démontrer la faisabilité de cette approche et de valider la pertinence des mesures proposées pour suivre l'évolution de la sécurité en opération[ODK99]. Depuis le développement du graphe des privilèges plusieurs travaux ont été menés sur le développement de formalismes à base de graphes pour décrire des vulnérabilités et des scénarios d'attaques (voir par exemple [PS98, SHJ+02, JNO03]). Par exemple, les arbres d'attaques « Attack trees » définis dans [Sch99] présentent une méthode structurée similaire aux arbres de fautes pour décrire différentes possibilités pour mener une attaque au sein d'un système. Des nouveaux résultats sur l'évaluation probabiliste de la sécurité ont été obtenus également par exemple sur l'évaluation de systèmes tolérants aux intrusions [GLR+03, WMT03], la modélisation de la propagation de malware basée sur des modèles épidémiologiques [SPW02, ZGT02, ZGTG05], ou bien l'utilisation de modèles plus récents, tels que la théorie des réseaux complexes issues de la physique ou bien la théorie des jeux [LW05, SHK06].

1.3.4.3 L'évaluation expérimentale

Les modèles décrits dans la section précédente s'appuient sur des hypothèses concernant le comportement des attaquants ou les vulnérabilités et les malveillances susceptibles d'affecter la sécurité des systèmes cibles. Ces hypothèses doivent être le plus proche possible de la réalité afin d'obtenir des mesures significatives. L'évaluation expérimentale peut fournir les données nécessaires pour élaborer ces hypothèses et valider les modèles correspondants. Elle regroupe les techniques et méthodes permettant de collecter des données caractérisant des activités malveillantes issues de l'observation opérationnelle du comportement des systèmes cibles ou bien basées sur des expériences contrôlées (pendant des phases de test). Les informations collectées doivent être choisies judicieusement pour aboutir à des analyses représentatives. De plus, un volume de données suffisamment grand et une longue période d'observation sont deux prérequis supplémentaires. De manière générale, l'évaluation expérimentale est réalisée en trois étapes. L'étape de collecte des données, à proprement parler, est tout d'abord réalisée. Pour ce faire, la manière de mener à bien cette collecte doit être réfléchie. Elle constitue le point clé pour les étapes suivantes. Ensuite, les données doivent être validées, afin d'éliminer celles qui sont erronnées ou qui ne sont pas représentatives des processus mis en jeu. A ce problème des données, vient se greffer le problème des données non observées, des données manquantes. Ces deux problèmes doivent être résolus avant toute analyse, pour fournir des données valides à l'étape suivante et ainsi pour ne pas aboutir à des résultats détachés de toutes représentations physiques. Quant à la dernière étape, elle permet d'obtenir les mesures de sûreté de

fonctionnement. Elle se base sur les données valides pour mener une analyse statistique. L'évaluation expérimentale dans le domaine de la sûreté de fonctionnement a toujours été un thème de recherche très actif. Historiquement, l'effort a d'abord porté sur l'étude de systèmes et de fautes du matériel. Des travaux ont ensuite été consacrés à la caractérisation du comportement en présence de fautes de logiciels et systèmes d'exploitation, de pilotes de périphériques, de systèmes distribués, des réseaux, de serveurs basés sur Internet et plus récemment sur des systèmes mobiles [SCK04]. La majorité de ces travaux s'intéressent aux fautes accidentelles. La caractérisation et la collecte de données permettant l'analyse du comportement de système en présence de malveillances a fait l'objet de plusieurs travaux durant la dernière décennie. Ces travaux incluent 1) le recensement des vulnérabilités connues dans des bases de données (par exemple Bugtrack[BUG], Cve[CVE], etc.), 2) le développement d'approches expérimentales pour l'analyse de l'efficacité de mécanismes de détection d'intrusions (voir par exemple les expériences du Lincoln Lab[LHF+00, McH00] ou les travaux présentés dans [DDW98, PZC+96]), 3) et le déploiement d'environnements pour la collecte de données caractérisant des attaques réelles sur Internet basés en particulier sur des sondes et des pots de miel. Les travaux présentés dans le cadre de cette thèse s'intéressent en particulier à ces dernières approches basés sur les pots de miel. La section suivante présente plus en détail les travaux connexes effectués sur ce thème.

1.4 Sondes et pots de miel

Dans cette section, nous présentons les sondes ainsi que des projets qui en exploitent le principe. Ensuite, nous présentons plus en détail les pots de miel, outils qui sont étudiés plus particulièrement dans le cadre de nos travaux.

1.4.1 Les sondes

Une sonde est un point de collecte de données sur Internet. La plupart du temps, elle repose sur la récupération des fichiers de journalisation des routeurs sur Internet ou des pare-feux. L'approche par sonde est très intéressante car elle a mis en évidence le caractère excessivement malin de plusieurs programmes sur Internet. Parmi les initiatives s'appuyant sur les sondes, nous pouvons citer les projets Caida[Cla01] et Dshield[DSH] qui sont présentés dans la section 1.4.3.

1.4.2 Les pots de miel

Les sondes abordées précédemment, présentent un inconvénient majeur. Les données collectées mélangent à la fois des données issues d'activités licites et des données issues d'activités illicites. Pour pallier ce problème, un outil à été créé : le pot de miel. Un pot de miel se définit comme un système informatique connecté à un réseau, volontairement vulnérable à une ou plusieurs failles et visant à attirer les attaquants afin d'étudier leur comportement. En théorie, aucune activité en provenance ou à destination de ce système ne devrait être enregistrée. Dans le cas contraire, il s'agit au mieux d'une erreur accidentelle, au pire d'une tentative d'attaque intentionnelle.

1.4.2.1 Historique des pots de miel

L'histoire des pots de miel remonte à la fin des années 80, lorsque le concept fut établi par Clifford Stoll dans [Sto88]. Dans ces travaux, l'auteur décrit les manières d'observer et pister un intrus. Pour ce faire, il a imaginé un projet gouvernemental factice, avec des informations factices, pour que les intrus passent un temps non négligeable à télécharger et analyser ces fichiers.

Dans les années 90, Cheswick implémenta et déploya un véritable pot de miel[Che91]. En parallèle à ces travaux, Bellovin [Bel92] a discuté des avantages et problèmes de telles implémentations. En 1998, Grundschober et Dacier ont introduit la notion de « détecteur renifleur »[Gru, GD98], une autre appellation pour le concept nommé aujourd'hui les « jetons de miel »(en anglais, *honeytoken*).

Les pots de miel, les jetons de miel et les technologies associées sont des notions utilisées depuis longtemps, même si la terminologie est récente. Ce n'est qu'en 2001 que le terme de pot de miel a été pour la première fois utilisé. Depuis, plusieurs auteurs ont proposé des définitions et des classifications. Une définition couramment employée est celle introduite par Lance Spitzner dans son livre intitulé « Honeypots, Tracking hackers »[Spi02] : *A honeypot is security resource whose value lies in being probed, attacked or compromised*. Depuis, le concept de pot de miel a évolué. Le document [PDD03] propose une terminologie et un aperçu des différentes technologies de pot de miel. Un réseau de miel (en anglais, *honeynet*) est un réseau de pots de miel. Un jeton de miel (en anglais, *honeytoken*) est un pot de miel inviolable. Un exemple de jeton de miel classique est décrit dans [Spi03b]. Ce dernier est représenté sous forme d'un enregistrement correspondant à un patient virtuel. Ce patient n'existant pas, aucun accès à cet enregistrement ne devrait être observé. Le cas contraire correspond à une violation des privilèges. La vitesse avec laquelle le domaine évolue a entraîné la création du projet *Honeynet Project*[Spi03a], dont le but est de constituer une base de connaissance sur les différentes technologies de pots de miel, les leçons reçues par leur emploi, les difficultés, etc.

Les pots de miel peuvent se décliner en plusieurs catégories, en fonction du niveau d'interaction fourni. Dans la littérature, une classification en deux catégories est utilisée. Elle distingue les pots de miel *basse interaction* des pots de miel *haute interaction*. Récemment, des travaux ont été réalisés afin de concilier les avantages de ces deux approches pour aboutir à des pots de miel dits *intermédiaires* ou *hybrides*.

1.4.2.2 Les pots de miel basse interaction

Les pots de miel basse interaction émulent des services réseaux pour tromper des programmes malveillants. Ces services ne peuvent donc pas être exploités pour obtenir un accès au système. Ils sont très répandus car simples à mettre en œuvre. De plus, du fait de l'émulation, ils permettent de bien contrôler les faits et gestes des attaquants.

Ce type de pot de miel permet de collecter des informations liées au dialogue entre le programme malveillant et le service. Ce dialogue est souvent plus court que la normale en raison de l'émulation. Les informations collectées concernent donc le début du dialogue. Toutefois, ces informations sont suffisantes pour identifier au plus tôt une activité malveillante.

Parmi les implémentations les plus simples, nous pouvons en citer les moniteurs

de ports tels que NukeNabber et Netcat[NET]. Ces outils permettent d'écouter sur un port et de journaliser les connexions. Les informations ne sont pas interprétées mais elles sont uniquement stockées : ils ne proposent aucune interaction avec les attaquants, De ce fait, seuls les premiers paquets des connexions sont récupérés, minimisant ainsi les risques. Ils sont utiles pour l'observation des vers.

Certaines implémentations, telles que Specter[SPE], Fakebo et Deception toolkit, proposent d'émuler des logiciels vulnérables : du système de transfert de fichier (`ftp`), au système de gestion des courriers électroniques (`pop`). Ils présentent aux attaquants les bannières de ces logiciels vulnérables. Ils permettent de récupérer les débuts des interactions avec les attaquants. Toutefois, ils ne trompent pas les attaquants humains, qui peuvent s'apercevoir très vite de la supercherie.

Honeyd[Pro04] est une implémentation qui s'est très vite imposée par sa simplicité et sa flexibilité d'utilisation. Il se présente sous la forme d'un démon qui émule des réseaux de machines. Il se configure par le biais d'un fichier contenant les informations sur le routage entre les machines et, pour chacune des machines émulées, le système d'exploitation émulé ainsi que les actions à associer à certains ports de la machine émulée. Préciser le système d'exploitation d'une machine émulée permet d'émuler le comportement de la pile TCP/IP du véritable système d'exploitation, en réponse à d'éventuelles « prises d'empreintes ». Quant aux scripts, ils permettent de préciser le comportement à adopter en fonction du dialogue qui s'installe entre l'attaquant et le service émulé sur le port en question.

L'outil Nepenthes[BKH+06] fournit un mécanisme intéressant pour traiter les attaques. Il simule des vulnérabilités afin d'inciter les attaquants à lui envoyer leurs maliciels. Il est constitué d'une base qui comprend des routines élémentaires de gestion des connexions, des fichiers, etc. Des modules peuvent lui être ajoutés afin de detecter les vulnérabilités utilisées et les logiciels employés.

1.4.2.3 Les pots de miel haute interaction

Les pots de miel haute interaction correspondent à un système réel ou à un système émulant un système complet. En émulant l'intégralité du système, le dialogue avec l'attaquant peut être plus important. Ce type de pot de miel permet, avant tout, de laisser les attaquants pénétrer le cœur du système. Ils constituent un excellent complément des pots de miel basse interaction. Leur inconvénient vient de l'attention importante à apporter au cloisonnement des activités des attaquants pour se protéger des risques d'utilisation du pot de miel pour attaquer d'autres machines (phénomène de rebond). Plusieurs implémentations de pots de miel ont été développées, par exemple *honeywall*[Pro05], *sebek*[Pro03], etc. La plupart se basent sur des systèmes d'exploitation à part entière.

Parmi ces implémentations, nous pouvons distinguer deux catégories. La première catégorie est constituée d'implémentations qui utilisent des outils tierces pour collecter des informations. Nous pouvons citer, à titre d'exemple, l'implémentation proposée dans [Hon03], qui utilise le renifleur de réseau snort afin de collecter les échanges entre le pot de miel et les attaquants. Le contrôle des connexions est réalisé avec un pare-feu. Une autre implémentation [HM03] propose d'utiliser un outil de detection d'intrusion, Prelude, pour la collecte des informations.

La seconde catégorie est constituée d'implémentations qui modifient le comporte-

ment du système d'exploitation pour y ajouter les mécanismes nécessaires à la collecte des données. SEBEK[Pro03] et UBERLOGGER[AGJ05] constituent des exemples de ces implémentations. Elles se basent sur des systèmes d'exploitation GNU-LINUX. Les noyaux de ces systèmes ont été modifiés afin d'y intégrer les mécanismes nécessaires pour intercepter les informations. Ces modifications ont aussi été apportées dans le but de ne pas être detectées par des attaquants. Une description plus détaillée de certaines de ces implémentations est fournie dans le chapitre 3.

1.4.2.4 Les pots de miel intermédiaires

Les pots de miel basse interaction proposent un faible niveau d'interaction avec les attaquants et les pots de miel haute interaction sont difficiles à mettre en œuvre en raison du risque qu'ils occasionnent. Le besoin de pots de miel intermédiaires ou hybrides s'est fait ressentir[BSHN04]. Certaines implémentations de pots de miel visent à utiliser ces deux types de pot de miel conjointement. Par exemple, dans [ASS+06], l'auteur propose une architecture basée sur le déployement conjoint de pots de miel basse interaction à base de HONEYD et de pots de miel haute interaction, dans un réseau local. Les systèmes d'exploitation installés sur ces derniers reflêtent la répartition des systèmes d'exploitation dans le réseau qui héberge les pots de miel. Toutes les connexions à destination des pots de miel basse interaction sont redirigées vers les pots de miel haute interaction.

Dans le même esprit, les travaux réalisés dans [BCW+04] proposent une architecture de pots de miel hybrides composées à la fois de pots de miel basse interaction et de pots de miel haute interaction. Les pots de miel basse interaction sont destinés à être déployés sur Internet. Les connexions qui leur sont destinées sont redirigées, de la même manière, vers les pots de miel haute interaction.

Dans [LMD05], les auteurs proposent d'utiliser, de la même manière, des pots de miel basse interaction et des pots de miel haute interaction. Le trafic malveillant à destination des pots de miel basse interaction et correspondant à de nouvelles activités est redirigé vers les pots de miel haute interaction. Le dialogue ainsi établi est analysé pour faire évoluer le niveau d'interaction des pots de miel basse interaction. Les pots de miel basse interaction possèdent une représentation de l'interaction des pots de miel haute interaction sous la forme d'automates.

1.4.3 Les projets basés sur les sondes et les pots de miel

Les outils et implémentations de sondes et pots de miel présentés dans les sections précédentes ont été utilisés dans des projets dans le but d'analyser les trafics malveillants. Dans cette section, nous présentons certains de ces projets. Une présentation plus complète peut se trouver dans [Pou06] et [PH07].

1.4.3.1 Les projets CAIDA et INTERNET MOTION SENSOR

Le projet CAIDA (Cooperative Association for Internet Data Analysis) a pour objectif d'analyser et comprendre les phénomènes globaux observés sur Internet. Pour mener des études du point de vue de la sécurité, des sondes particulières ont été développées dans le cadre de ce projet : les télescopes réseaux (en anglais, *network*

telescopes). Ils se définissent de la manière suivante : *un télescope réseau est une portion de l'espace des adresses IP routables pour lesquels peu voire aucun trafic légitime existe*[MSVS04].

L'observation à partir des télescopes réseaux a permis des études précises sur des vers tels que `slammer`[MPS+03] et `code-red`[MSB02] ainsi que sur le phénomène de *backscatter*[MSB+06]. Les données ayant permis ces analyses sont disponibles en ligne. CAIDA est un projet d'observation de haut niveau. Les résultats ainsi obtenus montrent l'importance de ce genre de dispositif.

L'INTERNET MOTION SENSOR est un projet de l'Institue du Michigan qui utilise un réseau de sondes distribuées. Ces sondes sont des adresses routables mais non utilisées. Elles sont au nombre de 2^{24}, soit 1/256 des adresses d'Internet. L'observation de ces sondes permet d'inférer des informations intéressantes sur les attaques et les attaquants, par exemple, les attaques de déni de service. Un des problèmes de cette approche est le côté passif des sondes. Effectivement, les sondes employées ne répondent pas aux paquets réseaux des attaquants.

1.4.3.2 Le projet DSHIELD

Le projet DSHIELD est un système de centralisation et d'analyse des fichiers de journalisation de différents pare-feux. L'utilité de cette base de données est donc principalement focalisée sur la sécurité.

Des volontaires sur Internet peuvent régulièrement envoyer leurs informations dans cette base de données pour l'enrichir. Actuellement, plus de 20 millions de fichiers de journalisation sont enregistrés chaque jour. Ils permettent d'élaborer des statistiques journalières sur les évolutions des tendances sur Internet. Ces données sont utilisées, par exemple, par le Internet Storm Center (ISC) pour l'élaboration de bulletins d'alerte. Elles permettent de suivre de manière très efficace la propagation des vers sur Internet. Toutefois, le caractère hétérogène des conditions dans lesquelles les données ont été collectées et le manque d'informations sur l'environnement de collecte rend difficile l'exploitation de ces données pour réaliser des analyses fines et non biaisées des activités des attaquants et de leur stratégie sur Internet.

1.4.3.3 Le projet Leurre.com

Le projet LEURRE.COM[PDP05] vise à déployer sur Internet une plate-forme basée sur les pots de miel basse interaction, dans un grand nombre d'endroits dans le monde. Les plates-formes déployées sont basées sur `HONEYD`. Elles collectent des données sur les activités malveillantes. Ces données sont centralisées dans une base de données hébergée et administrée par l'Institut Eurécom, à Sophia Antipolis. Ce projet se démarque des autres initiatives par l'utilisation de plates-formes homogènes de collecte de données configurées de façon identique. De la sorte, des analyses comparatives sur les activités observées en différents endroits dans le monde peuvent être réalisées sans être biaisées par la façon avec laquelle ces données sont collectées ni par les spécificités de chaque plate-forme.

Le projet LEURRE.COM a été mis en place dans l'optique de constituer un ensemble de données pouvant être partagé par la communauté scientifique pour effectuer différentes analyses et modélisations permettant d'améliorer nos connaissances sur les

activités malveillantes sur Internet et à contribuer à une meilleure compréhension des comportements et stratégies des attaquants. Le projet CADHO a été mis en place dans cette optique. Ce projet, financé dans le cadre de l'ACI Sésurité et Informatique, a servi de cadre pour les travaux effectués durant cette thèse. Ce projet regroupe trois partenaires : l'Institut Eurecom, le LAAS-CNRS et le CERT-Renater. Outre le développement et le déploiement de la plate-forme de pots de miel basse interaction, il vise deux objectifs principaux :

- Le développement de méthodologies d'analyse et de modèles pour caractériser les processus d'attaque observés à partir des données collectées sur les pots de miel basse interaction.
- Le développement de pots de miel haute interaction permettant de dépasser le stade d'analyse des attaques automatisées pour parvenir à comprendre et modéliser les stratégies mise en œuvre par des attaquants humains une fois qu'ils ont parvenu à compromettre une machine cible.

Dans [Pou06], l'auteur présente une méthode d'analyse appelée HoRaSis (*Honeypot tRaffic analySis*) qui a été développée pour faciliter l'acquisition de connaissances sur les malveillances, à partir des données sont issues de la plate-forme LEURRE.COM. Cette méthode est constituée de deux étapes. La première étape, dite de *discrimination*, permet de regrouper les activités observées qui partagent une *empreinte* similaire sur les pots de miel, en utilisant des techniques de classification et regroupement. Quant à la seconde étape, dite d'*analyse corrélative*, elle permet d'identifier des empreintes identiques. L'accent a été mis sur la manière de filtrer et regrouper les activités observées afin d'extraire des informations pertinentes sur les activités malveillantes, des similitudes de comportement sur plusieurs plates-formes de pot de miel ou des outils d'attaque utilisés. Des méthodes ont aussi été mises en place pour enrichir ces données par des informations telles que la localisation géographique des adresses à l'origine des activités malveillantes et les systèmes d'exploitation utilisés. Les résultats présentés dans ce manuscrit sont complémentaires à ces travaux et mettent l'accent sur le développement de modèles statistiques caractérisant les données issues des pots de miel basse interaction et, d'autre part, sur le développement et l'analyse de données collectées sur les pots de miel haute interaction.

1.5 Conclusion : orientation et contributions de la thèse

Dans ce chapitre, nous avons présenté le cadre général de nos travaux en mettant l'accent sur les activités malveillantes sur Internet, les motivations des pirates et les différents moyens nous permettant de faire face à ces utilisateurs malveillants. Nos travaux portent plus particulièrement sur le développement de méthodes et de modèles permettant d'améliorer notre connaissance sur les processus d'attaque ciblant des systèmes connectés au réseau Internet. L'évaluation expérimentale qui consiste à collecter des données issues de l'observation en opération des systèmes cibles constitue un moyen privilégié pour enrichir nos connaissances sur les vulnérabilités exploitées par les attaquants et sur les stratégies des intrus pour mettre en œuvre leurs attaques.

Plusieurs études ont été menées sur ce sujet durant la dernière décennie en utilisant les pots de miel comme support de base pour observer et collecter des données sur les attaquants. En effet, il existe actuellement plusieurs plates-formes et environnements

de collecte de données fournissant des informations intéressantes sur les attaques sur Internet. Plusieurs travaux existent dans la littérature sur l'exploitation des données issues de pots de miel basse interaction pour caractériser les activités malveillantes. Des avancées significatives ont été obtenues dans ce domaine et se sont concrétisées par le développement de méthodologies permettant d'extraire de façon semi-automatique des informations pertinentes sur les activités d'attaques observées. Ces études ont aussi montré que les données collectées par les pots de miel basse interaction peuvent fournir des informations très intéressantes à condition d'utiliser des méthodologies d'analyse rigoureuses. Les travaux présentés dans ce manuscrit s'inscrivent dans cette logique et sont complémentaires aux résultats obtenus dans ce domaine.

Nos travaux s'inscrivent dans le cadre du projet Cadho. Tout d'abord, nous nous appuyons sur les données collectées à partir de la plate-forme Leurre.com mise en place et administrée par l'Institut Eurecom, dont le déploiement a été initié dans le cadre du projet Cadho. Cette plate-forme, opérationnelle depuis plus de quatre ans, offre un volume important de données d'attaques recueillies sur un ensemble de pots de miel basse interaction, identiquement configurés, qui ont été déployés à différents endroits du globe.

Les principales contributions novatrices de nos travaux concernent :

- Le développement d'une méthodologie permettant l'élaboration de modèles stochastiques caractérisant les processus d'occurrence des attaques et leur propagation, à partir des données issues de plusieurs plates-formes de pot de miel basse interaction déployées sur Internet.
- L'instrumentation et le déploiement d'un pot de miel haute interaction qui nous a permis d'observer des attaques manuelles et d'analyser les activités effectuées par des intrus une fois qu'ils ont réussi à prendre le contrôle d'une machine cible. De telles analyses ne sont pas possibles avec des pots de miel basse interaction.

A notre connaissance, l'élaboration de modèles stochastiques décrivant les processus d'attaque observés à partir des données collectées via des pots de miel basse interaction a fait l'objet de peu de travaux jusqu'à maintenant. De tels modèles sont utiles pour générer des traces représentatives du trafic malveillant observé sur Internet dans le but de valider des systèmes et des mécanismes de sécurité. Ils sont aussi utiles pour élaborer des hypothèses et des distributions réalistes des processus d'attaque, qui sont nécessaires pour les travaux visant à faire des évaluations prévisionnelles de la sécurité.

Peu de travaux existent également sur le développement et l'exploitation de pots de miel haute interaction pour analyser le comportement des attaquants une fois qu'ils ont réussi à entrer au cœur du système. L'architecture de pot de miel haute interaction proposée dans ce manuscrit et les résultats des analyses des données collectées à partir de son déploiement apportent des enseignements sur la faisabilité de ce type d'expérimentation, la complémentarité avec les travaux menés sur les pots de miel basse interaction et les difficultés qui restent à résoudre.

Outre ces contributions, nous avons aussi développé des méthodologies pour guider l'exploitation des données issues des pots de miel basse interaction et du pot de miel haute interaction en fonction des analyses menées.

Ces résultats sont développés dans les chapitres suivants.

Chapitre 2

Caractérisation des processus d'attaques à partir des pots de miel basse interaction

La collecte de données et le développement de méthodes permettant l'analyse des activités malveillantes sur Internet (virus, vers, attaques de déni de service, etc.) sont importants pour comprendre les mécanismes qui régissent les processus d'attaque et extraire des informations pertinentes pouvant être utiles pour guider la conception, la validation, l'évaluation et l'exploitation opérationnelle des systèmes. Différents types d'analyses peuvent être effectuées selon les objectifs visés. Ces analyses peuvent être de nature qualitative basées sur des statistiques descriptives permettant par exemple d'identifier les principales vulnérabilités ciblées, l'origine des attaques, les outils utilisés par les attaquants, etc. Elles peuvent également avoir pour objectif d'identifier des modèles stochastiques et des distributions de probabilités qui soient représentatifs des processus d'occurrence des attaques.

Les travaux présentés dans ce chapitre portent sur le développement d'une méthodologie utilisant des outils statistiques et mathématiques pour caractériser le processus d'occurrence des attaques à partir des données collectées par des pots de miel basse interaction déployés sur Internet. Nous nous intéressons particulièrement à l'élaboration de modèles décrivant la distribution des intervalles entre attaques observés sur divers environnements de pots de miel, la propagation des attaques entre environnements, et des corrélations observées entre les évolutions des fréquences d'occurrence des attaques sur différents environnements, toutes sources confondues ou en fonction de leur origine géographique. Nous utilisons les données recueillies par la plate-forme Leurre.com mise en place et administrée par l'Institut Eurécom comme base et source d'information pour développer notre méthodologie et l'illustrer.

Ce chapitre suit la structure suivante. Nous présentons d'abord l'architecture des pots de miel et de plate-forme de collecte de données Leurré.com. Nous présentons ensuite une vue globale des données utilisées dans le cadre de notre étude et des problèmes qu'il est nécessaire de résoudre dans l'optique d'utiliser ces données pour élaborer des modèles des processus d'attaque observés. La méthodologie proposée pour résoudre ces problèmes est présentée d'abord de façon succincte puis développée dans les paragraphes suivants. La méthodologie est illustrée sur les données collectées

au fur et à mesure de la présentation de ses différentes étapes.

2.1 Architecture des pots de miel et base de données

Dans cette section, nous présentons l'architecture de pot de miel basse interaction développée et déployée par l'Institut Eurécom, ainsi que les données collectées.

2.1.1 Architecture

L'architecture déployée dans le cadre du projet LEURRE.COM se base sur des pots de miel basse interaction[PDP05]. Elle est présentée à la figure 2.1. Les systèmes d'exploitation qui sont simulés ont été choisis de manière à reflèter au mieux leur répartition sur Internet. Le but de cette démarche est de pouvoir observer des comportements des attaquants dépendant de la cible. Trois systèmes sont proposés aux attaquants : RED HAT 7.3, WINDOWS 98 et WINDOWS NT. Afin d'en faciliter le déploiement, l'administration et la collecte des données, ces systèmes sont simulés avec le programme HONEYD[Pro04]. Ce dernier ne nécessite que peu de ressources pour fonctionner. Ainsi, cette architecture peut être déployée avec une machine peu onéreuse et donc avec un faible investissement de la part des partenaires. Les paquets échangés entre les attaquants sur Internet et les différents systèmes d'exploitation sont stockés dans une base de données par l'intermediaire du logiciel TCPDUMP. Cette architecture est alors déployée chez les partenaires du projet. Nous nommons *environnement* un déploiement de cette architecture chez un des partenaires. Les données collectées par les différents environnements sont récupérées régulièrement pour être stockées dans un unique serveur de base de données. Une interface web permet alors d'interroger cette base pour obtenir des informations intéressantes.

Les services disponibles sur les systèmes simulés sont des services courants (`ftp`, `web`, etc.). La cohérence entre système et service est préservée. Un service qui est normalement absent sur un système n'est pas installé sur le système simulé correspondant. Par exemple, le service `ssh` n'est pas disponible sur le système WINDOWS 98 simulé. Les attaquants peuvent dialoguer avec ces services. Les connexions entrantes sont autorisées. Rappelons que HONEYD permet la mise en place d'un pot de miel basse interaction. Les attaquants ne peuvent pas, a priori, rentrer au cœur du système. Toutefois, pour se prémunir de tout débordement non contrôlé, le pare-feu a été configuré de manière à interdire les connexions sortantes.

Un avantage de cette démarche, par rapport aux autres projets, est lié au déploiement d'une architecture homogène. De la sorte, les données collectées proviennent d'environnements qui sont comparables. Les résultats, environnement par environnement, peuvent être confrontés pour obtenir des résultats significatifs. Nous présentons dans la suite ces données.

2.1.2 La base de données

L'unité d'observation offerte par les environnements est le paquet réseau. Un paquet réseau p est caractérisé par l'adresse de la machine à l'origine de la connexion $adr(p)$, le système ciblé $sys(p)$ (c'est-à-dire la machine virtuelle), l'environnement ciblé $env(p)$ et la date de réception $date(p)$. L'ensemble des paquets observés est noté

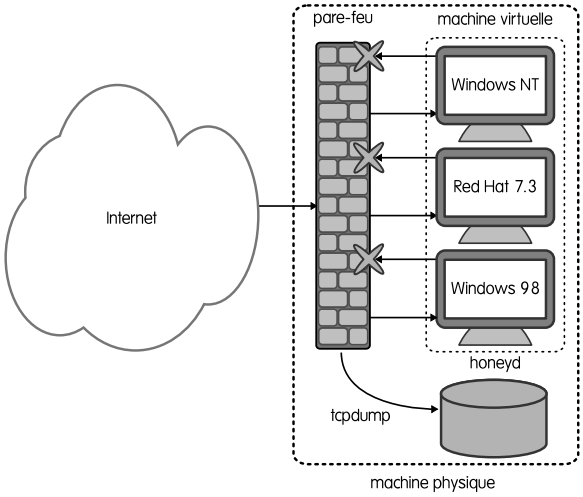


Fig. 2.1 – Architecture du pot de miel basse interaction, projet Leurre.com

Ps. Une machine d'Internet peut communiquer avec les environnements de collecte. Le nombre de paquets échangés lors d'une communication est très variable. Mener des analyses de haut niveau sur l'ensemble des paquets peut très vite devenir fastidieux.

Nous pouvons considérer trois niveaux différents pour l'analyse des attaques : (1) l'ensemble des environnements de pots de miel déployés sur Internet, (2) un environnement de pot de miel particulier ou (3) un système donné d'un environnement de pot de miel. Pour faciliter ces analyses selon ces différents niveaux, il est nécessaire de regrouper les paquets de façon à définir des sessions d'attaque cohérentes avec chacun de ces niveaux. Les définitions de *Source*, *Global_Session*, *Large_Session* et *Tiny_Session* présentées dans [Pou06] ont été introduites dans cet objectif.

- Une Source correspond à une adresse IP observée sur un ou plusieurs environnements, et pour laquelle le temps d'arrivée entre deux paquets consécutifs reçus reste inférieur à un certain seuil (25 heures). La différence de temps se calcule en convertissant toutes les dates au format GMT (temps moyen de Greenwich, en anglais *Greenwich Mean Time*).
- Une Global_Session est l'ensemble de paquets qui ont été échangés entre une Source et tous les environnements de pots de miel du projet Leurré.com.
- Une Large_Session est l'ensemble de paquets qui ont été échangés entre une Source et un environnement de pot de miel particulièr.
- Une Tiny_Session est l'ensemble de paquets qui ont été échangés entre une Source et une machine virtuelle donnée. Comme chaque environnement de pot de miel émule trois machines virtuelles, une Large_Session est composée d'au plus 3 Tiny_Session.

Ces définitions ont donné lieu à des routines de pré-traitement lancées régulièrement sur les données. Elles permettent de mettre à jour les ensembles définis précédemment. Les travaux présentés dans ce chapitre se basent sur ces données pré-traitées. Etant donné que nous sommes intéressés par le comportement des attaquants lorsqu'ils ciblent un site informatique dans son ensemble, nous étudierons plus particulièrement les *Large_Sessions*. Nous définissons une *Large_Session* comme étant une attaque ou

encore une session.

Pour mener des analyses statistiques sur la base de ces données, nous avons besoin d'une méthodologie. De manière générale, deux phases distinctes permettent de mener une analyse de données : la *phase exploratoire* et la *phase confirmatoire*[Ber86]. La phase exploratoire est tout d'abord nécessaire pour fortifier l'intuition. Elle se base sur une description des données et sur des représentations visuelles. La phase confirmatoire permet de confirmer les hypothèses, par des outils statistiques. La section suivante constitue la phase exploratoire. Nous y présentons des analyses préliminaires des sessions. A l'issue de cette section, nous présentons la méthodologie pour mener à bien la phase confirmatoire.

2.2 Vue globale et analyses préliminaires des données

En considérant les informations disponibles directement à partir de la base de données Leurre.com, actuellement, plus de 4 ans de collecte sont recensés dans la base de données pour un total de 4 873 564 sessions et 3 026 962 adresses différentes. La base de données recense également 80 environnements de pots de miel. Les environnements ont été déployés au fur et à mesure de l'avancée du projet. Ces données sont présentées dans cette section.

2.2.1 Présentation et analyse préliminaire

Les dates de mise en service des environnements sont différentes, comme indiqué sur la figure 2.2. L'axe des abscisses de ce graphique représente le temps écoulé entre février 2003 et août 2007. Les ordonnées représentent les environnements, ordonnés par la date de la première activité enregistrée. Des rectangles sont reportés sur chacune des lignes. Ils représentent le temps écoulé entre la première et la dernière activité observées sur les environnements correspondants. Cette durée est nommée la *durée d'activité*.

env. k	$\#LS_i$	$\#adr$	$\#LS_i/\#adr$
1	61305	57077	1.07
2	27653	25123	1.10
4	59935	52315	1.14
5	75898	70887	1.07
6	504652	235781	2.14
8	95565	57797	1.65
9	220759	195398	1.12
10	54415	51726	1.05
11	65465	55480	1.17
12	0	0	—
13	45659	41338	1.10
14	83778	65298	1.28
20	314381	230128	1.36
21	171036	147146	1.16
22	17604	13375	1.31
23	71091	64304	1.10
24	11395	8161	1.39
25	167608	130504	1.28
26	79518	67218	1.18
27	194438	169819	1.14
28	13248	9028	1.46
29	0	0	—
30	21995	14936	1.47
31	182301	159381	1.14
32	100657	85335	1.17
34	89321	81288	1.09
35	26617	25379	1.04

env. k	$\#LS_i$	$\#adr$	$\#LS_i/\#adr$
36	9089	8285	1.09
37	673	507	1.32
38	1845	1835	1.00
39	15367	12200	1.25
40	28133	22989	1.22
41	65847	58043	1.13
42	40803	32342	1.26
43	12326	9893	1.24
44	41153	36906	1.11
45	77574	68859	1.12
46	4	3	1.33
47	11103	6299	1.76
48	63067	56462	1.11
49	83008	70497	1.17
50	221181	151494	1.45
51	17739	16699	1.06
52	6030	3499	1.72
53	72902	65558	1.11
54	86493	75267	1.14
55	52768	47731	1.10
56	105238	79275	1.32
57	268923	210379	1.27
58	10417	8945	1.16
59	60994	54020	1.12
60	38388	28873	1.32
61	47065	41172	1.14
62	26100	20400	1.27

env. k	$\#LS_i$	$\#adr$	$\#LS_i/\#adr$
63	47694	41593	1.14
64	72986	55603	1.31
65	41563	37280	1.11
66	10795	9203	1.17
67	12303	9794	1.25
68	15150	12567	1.20
69	21519	19345	1.11
70	4766	4011	1.18
71	58326	42159	1.38
72	21178	17374	1.21
73	17604	14958	1.17
74	26908	22615	1.18
75	30075	26350	1.14
76	20321	17180	1.18
77	80566	59369	1.35
78	17379	12690	1.36
79	50999	32132	1.58
80	27370	24759	1.10
81	7931	6387	1.24
82	15663	15061	1.03
83	14570	13115	1.11
84	19285	16937	1.13
85	3029	2496	1.21
86	6	6	1.00
87	14437	10851	1.33
88	617	518	1.19

TAB. 2.1 – Nombre de sessions et d'adresses par environnement

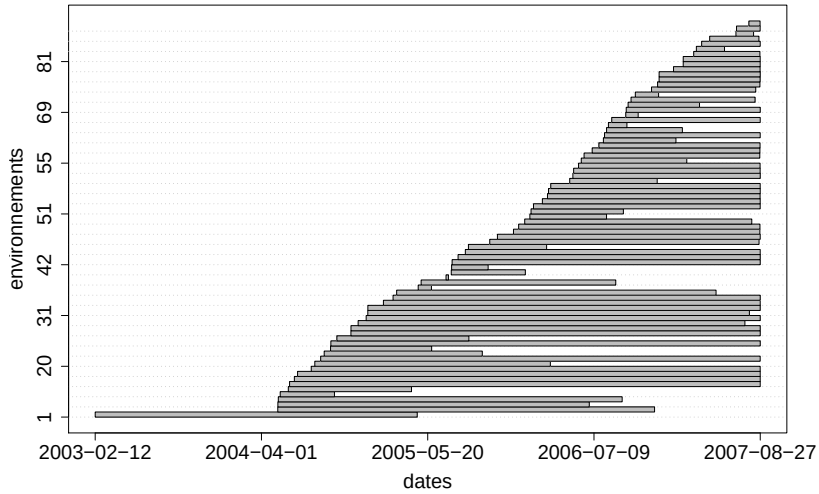


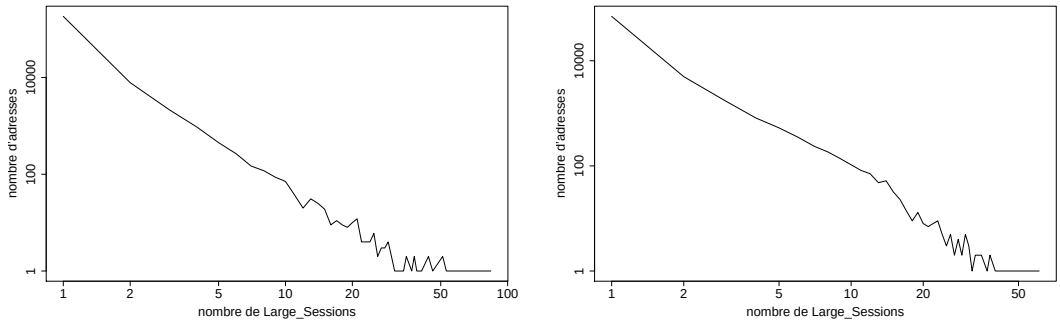
FIG. 2.2 – Evolution du nombre d'environnements déployés sur LEURRÉ.COM et durée d'activité pour chacun

La première remarque concerne le premier environnement. C'est celui qui a servi pour réaliser les premières expérimentations avant d'envisager un déploiement à plus grande échelle sur Internet. Ensuite, nous remarquons que certains environnements ont été déployés depuis peu de temps. Le nombre de jours de collecte est très faible pour ces environnements. Pour finir, nous constatons que plusieurs environnements ont été actifs pendant quelques jours uniquement.

Le tableau 2.1 résume l'activité observée sur chacun des environnements durant toute la période d'observation. La première colonne représente l'environnement. Les numéros des environnements correspondent à ceux utilisés dans la base de données. La seconde colonne contient le nombre de sessions de l'environnement. La troisième colonne contient le nombre d'adresses différentes ayant ciblé l'environnement. La dernière colonne correspond au rapport entre le nombre de sessions et le nombre d'adresses différentes. Certains environnements ont eu une activité soutenue et d'autres ont eu une activité très faible. Certaines lignes sont grisées. Elles correspondent à des environnements pour lesquels l'activité est négligeable. Ces environnements sont ignorés dès à présent, ramenant le nombre d'environnements restants à 76.

Nous pouvons constater que le rapport entre le nombre de sessions et le nombre d'adresses différentes est proche de 1. Le nombre d'adresses observées sur chacun des environnements est du même ordre de grandeur que le nombre de sessions. Pour la majorité des adresses, le nombre de sessions observées est faible. Peu d'adresses ont été très actives. Ceci est illustré par l'exemple sur la figure 2.3, pour les environnements 9 et 56. L'axe des ordonnées indique le nombre d'adresses pour lesquelles nous avons enregistré le nombre de sessions observées indiqué par l'axe des abscisses. Les échelles sont logarithmiques. La tendance linéaire décroissante montre que la distribution décrivant le nombre de *Large_Sessions* réalisées par adresse suit une loi polynomiale décroissante (en anglais, *power law*). Ce type de distribution a aussi été observé dans

des analyses de trafics web ou dans des études économétriques.



(a) environnement 9 (b) environnement 56

Fig. 2.3 – Nombre d'adresses, en ordonnée, ayant réalisé le nombre de sessions correspondant

Cette constatation nous amène à considérer les adresses très actives. Les tableaux de la figure 2.4 présentent les adresses ayant réalisé le plus d'activités. Le tableau 2.4(a) référence les 10 adresses ayant réalisé le plus de sessions. La deuxième colonne contient le nombre de sessions. La troisième colonne contient le nombre d'environnements ciblés par ces adresses. La dernière colonne contient le nom de l'environnement le plus ciblé par ces adresses avec, entre parenthèses, le nombre de sessions associé. Le tableau 2.4(b) constitue le complémentaire du précédent tableau. Il référence les 10 adresses ayant ciblé le plus d'environnements. De la même manière, la deuxième colonne contient le nombre d'environnements ciblés. La troisième colonne contient le nombre de sessions réalisées. La dernière colonne contient le nom de l'environnement le plus ciblé par ces adresses avec, entre parenthèses, le nombre de sessions associé.

adr. j	$\#LS_i$	#env.	env. max k ($\#LS_k$)
1	982	37	13 (59)
2	829	31	11 (63)
3	705	40	48 (37)
4	700	46	9 (37)
5	693	43	9 (34)
6	566	45	27 (24)
7	529	43	14 (29)
8	480	45	14 (33)
9	457	45	59 (24)
10	442	43	9 (25)

(a) en terme de nombre de sessions

adr. j	#env.	$\#LS_i$	env. max k ($\#LS_k$)
11	55	242	14 (14)
12	55	343	31 (25)
13	54	198	32 (9)
14	54	309	13 (23)
15	53	256	11 (18)
16	53	241	8 (15)
17	49	210	44 (11)
18	49	258	31 (14)
19	49	305	9 (21)
20	48	337	6 (17)

(b) en terme de nombre d'environnements

Fig. 2.4 – Liste des adresses ayant réalisé les plus fortes activités

Au vue de ces deux tableaux, on peut remarquer que les adresses ayant réalisé le plus de sessions ne sont pas celles ayant ciblé le plus d'environnements, et inversement. Autrement dit, un environnement n'est ciblé que par un sous ensemble des adresses

d'Internet, et pas forcément le même que celui d'un autre environnement. De plus, pour chacune des adresses, le nombre de sessions réalisées sur leur environnement privilégié est relativement faible comparé au nombre total de sessions. Nous constatons en général que les sessions des adresses les plus actives sont relativement bien réparties sur l'ensemble des environnements qu'elles ont ciblé.

2.2.2 Problème lié aux périodes de silence suspectes

Comme tout système informatique, les environnements peuvent connaître des périodes d'indisponibilité plus ou moins longues. Leur disponibilité est tributaire aussi de l'état du réseau l'accueillant. A titre d'exemple, le soir du samedi 4 novembre 2006, en raison d'incidents sur le réseau électrique allemand, des millions de foyers en Allemagne, Espagne, France et Italie, entre autres, ont été privés de courant[GdRdTd]. La coupure a aussi rendu indisponibles les environnements installés dans les régions affectées. Ces indisponibilités se traduisent par des attaques non observées et, au niveau des données, elles entraînent de longues périodes d'inactivités. Que penser d'un long silence soudain des attaquants sur un environnement, alors qu'habituellement des milliers d'entre eux s'acharnent sur lui ?

L'intervalle indiqué sur la figure 2.2 pour chaque environnement, qui correspond au temps écoulé entre la date de la première session et de la dernière, peut aussi inclure des périodes plus ou moins longues pendant lesquelles aucune activité malveillante n'a été enregistrée. De telles périodes peuvent être dues à des coupures d'electricité ou à des problèmes d'inaccessibilité de l'environnement correspondant.

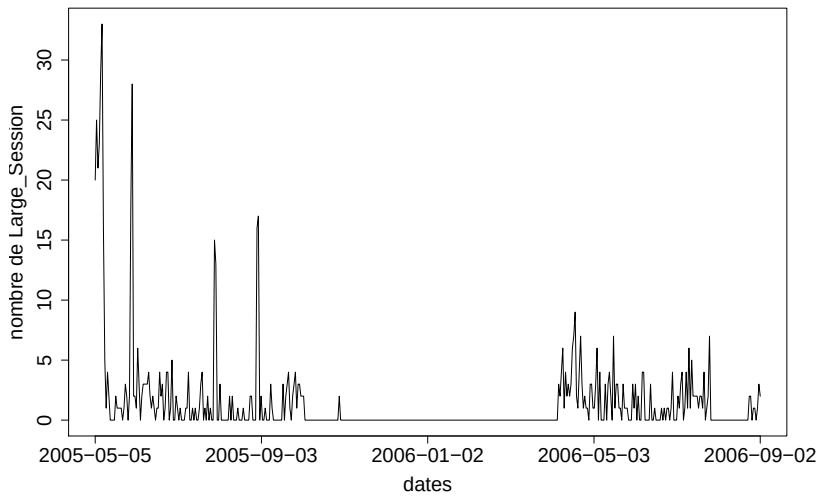


Fig. 2.5 – Evolution du nombre de sessions observées par jour sur l'environnement 37

A titre d'illustration, la figure 2.5 montre l'évolution du nombre de sessions observées par jour sur l'environnement 37. Nous pouvons constater que, entre novembre 2005 et avril 2006, aucune attaque n'a été enregistrée. Nous ne pouvons que difficilement envisager que cette période soit due à un silence des attaquants à l'égard de cet environnement. Nous sommes plutôt enclin à penser que cette longue période de

silence de 5 mois est due à une indisponibilité.

2.2.3 Discussion

Ces analyses de haut niveau montrent que les attaques observées sur Internet suivent des processus compliqués et bruités par des indisponibilités. Nous avons besoin d'outils statistiques pour mieux caractériser ces processus d'attaque et faire des analyses comparatives.

Notre objectif est d'élaborer des modèles des processus d'attaque qui soient représentatifs des processus réels, à partir des données observées. Il est donc nécessaire d'identifier les périodes d'indisponibilité pour atténuer leur impact sur les résultats et de sélectionner les données qui sont utilisées pour l'élaboration de modèles. La stratégie d'analyse, proposée dans la suite, procède en deux étapes. La première étape est basée sur l'hypothèse que les périodes d'indisponibilité, qui risquent d'affecter la validité des résultats, correspondent à des valeurs qui s'écartent de façon significative de la dynamique du processus d'attaque. Ces périodes, appelées *valeurs aberrantes*, peuvent être estimées en utilisant des outils statistiques. Nous appellerons par la suite *période de silence suspecte*, une période que nous supposons être une période d'indisponibilité. La deuxième étape consiste à sélectionner la plus grande période d'observation pendant laquelle un nombre important d'environnements ont été disponibles. Les données correspondant à cette période nous serviront à l'élaboration des modèles des processus et à analyser de façon comparative les comportements observés sur différents environnements.

2.3 Méthodologie d'analyse

Dans cette section, nous définissons les notations que nous utilisons. Les données étant issues d'un processus d'expérimentation, elles sont bruitées. Nous précisons alors quelles sont les variables qui nous intéressent en tenant compte du bruit des données dû aux périodes de silence suspectes. Ensuite, nous exposons la méthodologie d'analyse des données.

2.3.1 Notations et définitions

Chaque session, notée LS_i, possède plusieurs caractéristiques. Nous pouvons citer, entre autres, l'adresse de l'attaquant ($adr(LS_i)$), sa localisation géographique ($geo(LS_i)$), l'environnement ciblé ($env(LS_i)$), la date du début de l'activité ($datedebut(LS_i)$) et la date de fin de l'activité ($datefin(LS_i)$).

$$adr(LS_i) = adr(p)/p \in LS_i \tag{2.1}$$
$$geo(LS_i) = geo(adr(p))/p \in LS_i \tag{2.2}$$
$$env(LS_i) = env(p)/p \in LS_i \tag{2.3}$$
$$datedebut(LS_i) = \min\{date(p)/p \in LS_i\} \tag{2.4}$$
$$datefin(LS_i) = \max\{date(p)/p \in LS_i\} \tag{2.5}$$

Les sessions représentent les activités des attaquants sur les environnements de pot de miel pendant leurs attaques. Le processus à leur origine correspond au comportement des attaquants. Le comportement est alors observé à travers les données collectées sur les pots de miel. Soit $X_k(t)$ un processus stochastique binaire indiquant si une session a démarré à l'instant t ($X_k(t) = 1$) ou non ($X_k(t) = 0$), sur le pot de miel k. Soit $\tau_{k,i}$ l'instant de la i-ème session de $X_k(t)$. Les valeurs des $\tau_{k,i}$ permettent de déterminer les durées séparant deux sessions consécutives. Soit $T_{k,n}$ la durée séparant les $(n-1)$-ième et n-ième sessions. Ces différentes variables sont liées de la manière suivante :

$$\tau_{k,i} = \begin{cases} 0, & \text{si } i = 0 \\ \min(t/t > \tau_{k,i-1} \wedge X_k(t) = 1), & \text{sinon} \end{cases} \tag{2.6}$$

$$T_{k,n} = \tau_{k,n} - \tau_{k,(n-1)} \tag{2.7}$$

Idéalement, nous voudrions observer $X_k(t)$ pour mener nos analyses. Or, Les dispositifs d'observation ne sont pas parfaits. Concernant les environnements, nous avons identifié une source d'imperfection : les périodes de silence suspectes.

Le processus réellement observé sur l'environnement k n'est donc pas $X_k(t)$ mais $X_k(t) \cdot Up_k(t)$. La variable $Up_k(t)$ représente la disponibilité de l'environnement k. Elle prend ses valeurs dans l'ensemble $\{0; 1\}$. La valeur 1 indique que l'environnement n'est pas dans une période de silence suspecte et la valeur 0 indique qu'il est dans une période de silence suspecte. $Up_k(t)$ joue donc le rôle de filtre :

$$Up_k(t) = \begin{cases} 0, & \text{l'environnement } k \text{ est dans une période de silence suspecte} \\ 1, & \text{l'environnement } k \text{ n'est pas dans une période de silence suspecte} \end{cases} \tag{2.8}$$

Autrement dit, nous observons $X_k(t)$ uniquement lorsque $Up_k(t) = 1$. Par conséquent, en absence de surveillance continue de la disponibilité des environnements, nous ne sommes pas en mesure de déterminer si, entre deux sessions observées consécutives, des sessions n'ont pas été observées. La séquence des dates des sessions observées, notée $\tau'_{k,i}$ n'est donc pas la même que la séquence des dates des sessions, $\tau_{k,i}$, qui sont initiées par les attaquants. La différence correspond à des tentatives d'attaque qui ne sont pas observées à cause de la présence de périodes de silence suspectes sur l'environnement ciblé.

$$\tau'_{k,i} = \begin{cases} \min\{\tau_{k,j}/Up_k(\tau_{k,j}) = 1\}, & \text{si } i = 1 \\ \min\{\tau_{k,j}/Up_k(\tau_{k,j}) = 1 \wedge \tau_{k,j} > \tau'_{k,(i-1)}\}, & \text{sinon} \end{cases} \tag{2.9}$$

A ce problème, vient se rajouter l'arrondi sur les données : les dates sont arrondies à la seconde inférieure. La granularité que nous considérons est donc la seconde. Ainsi, nous observons $\tau'_{k,i}$ mais nous traitons $\lfloor \tau'_{k,i} \rfloor$. Le temps entre deux sessions consécutives que nous considérons est noté $T'_{k,n}$, où $T'_{k,n} = \lfloor \tau'_{k,n} \rfloor - \lfloor \tau'_{k,(n-1)} \rfloor$. Nous noterons par $X'_k(t)$ le processus stochastique observé sur l'environnement k. La figure 2.2 résume ces notations. Quant à la figure 2.6, elle représente les différentes variables utilisées.

Attaquants	Pot de miel
$X_k(t)$	$X'_k(t)$
$\tau_{k,i}$	$\lfloor \tau'_{k,i} \rfloor$
$T_{k,n}$	$T'_{k,n}$

TAB. 2.2 – Variables décrivant les processus d'attaque

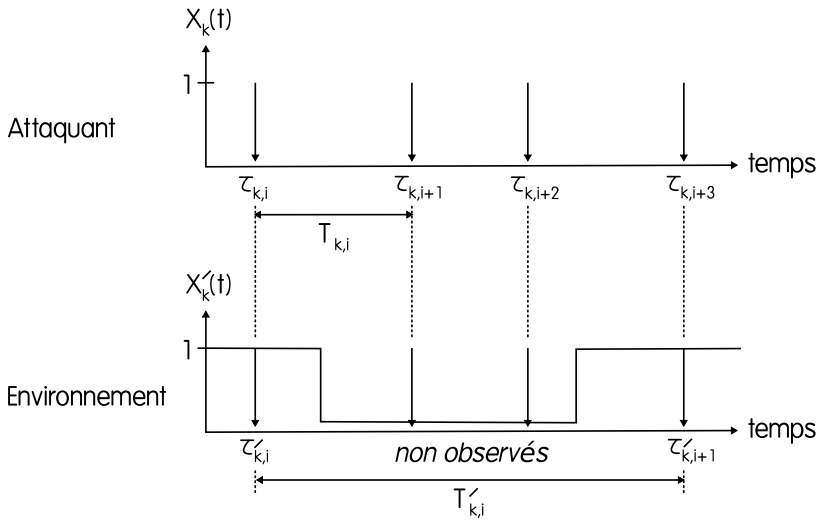


FIG. 2.6 – Variables représentant les dates des sessions et les durées entre deux sessions consécutives

2.3.2 Méthodologie

Notre objectif est d'exploiter les données collectées sur les pots de miel basse interaction pour construire des modèles permettant de caractériser les processus d'attaque sur Internet : la modélisation de l'intervalle entre attaque, l'évolution du nombre d'attaques par jour en fonction de l'origine géographique des attaquants et la propagation des adresses sur les différents environnements.

Afin de réaliser des analyses comparatives sur la base de ces modèles, il est nécessaire de considérer des environnements pour lesquels la période d'observation a été suffisamment longue et sans trop de périodes de silence suspectes. De plus, il est important d'analyser les comportements observés sur différents environnements pendant la même période calendaire. Ceci nécessite un prétraitement des données pour satisfaire ces critères.

Nous avons donc établi une méthodologie en cinq étapes, présentée à la figure 2.7. Nous avons, dans la section précédente, présenté les données afin d'identifier le problème de périodes silence suspectes. A partir de ce constat, nous pré-traitons ces données afin d'identifier les valeurs aberrantes liées à ces périodes, pour chacun des environnements. Cette identification nous permet de sélectionner une partie des données pour laquelle l'impact des valeurs aberrantes est minime. Sur la base de cette période, nous menons les trois analyses. Ce présent chapitre est structuré de manière à refléter la méthodologie. Ainsi, la section suivante présente le prétraitement sur les

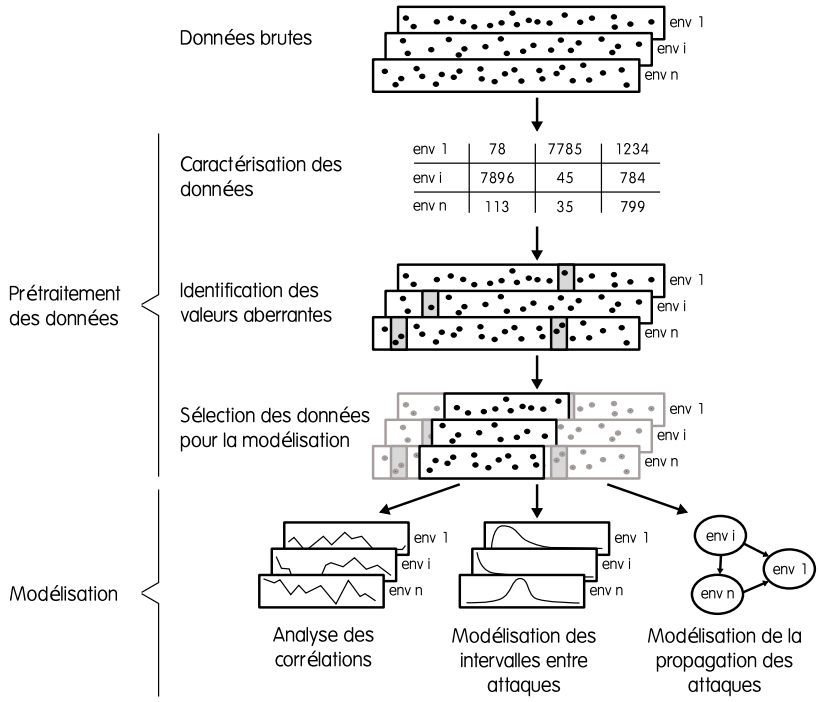


FIG. 2.7 – Processus de traitement des données collectées sur les environnements de pot de miel déployés sur Internet

données.

2.4 Prétraitement des données

Le prétraitement des données consiste en particulier à identifier et traiter les périodes de silence suspectes des environnements. Ces longues périodes d'inactivité, susceptibles d'affecter la validité des résultats, ont été abordées dans la section 2.2.2. Elles peuvent être considérées comme des valeurs aberrantes. N'étant pas toujours en mesure d'observer directement l'état des pots de miel, ces longues périodes sont difficiles à identifier. Nous allons donc profiter des travaux réalisés dans le domaine de la détection des valeurs aberrantes pour identifier les périodes de silence suspectes.

2.4.1 Méthodes d'identification des valeurs aberrantes

Lors d'une expérimentation, il est fréquent que des données s'écartent de façon significative des autres. Suivant les cas, ces valeurs peuvent être considérées comme normales mais peu fréquentes ou alors comme aberrantes. Une définition de valeur aberrante a été donnée par Grubbs en 1969[Gru69] : [une valeur aberrante est] *une observation qui semble dévier de façon marquée par rapport à l'ensemble des autres membres de l'échantillon dans lequel elle apparaît.* Les valeurs aberrantes sont classées en trois catégories : les valeurs aberrantes d'amplitudes, les valeurs aberrantes spatiales et les valeurs aberrantes relationnelles. Dans cette section, nous ne traitons que des valeurs aberrantes d'amplitudes. Elles sont des valeurs excessivement élevées

par rapport aux autres valeurs. Des valeurs aberrantes d'amplitudes sont des valeurs extrêmes qui sont peu fréquentes. L'inverse n'est pas vrai. Des valeurs extrêmes rares ne sont pas forcément des valeurs aberrantes.

Les valeurs aberrantes doivent être traitées avec précaution, sous peine de biaiser les résultats. Pour fixer les idées, considérons l'ensemble des valeurs $\{10, 10, 12, 7, 8, 8, 9, 11, 6, 7\}$. La moyenne de cet ensemble est $8,8$. Maintenant, admettons que la valeur aberrante 42 s'insère dans les données. Cette valeur diffère de manière significative de la tendance de l'ensemble. La moyenne se trouve alors fixée à $11,8$. Une seule valeur aberrante peut avoir un impact important sur les résultats. Ignorer ce problème peut donc biaiser les analyses statistiques.

Divers tests statistiques ont été présentés dans la littérature pour la détection des valeurs aberrantes[Pla05, HA04]. Citons, entre autres, les test de Nixon, de Grubbs et de *boxplot*. Dans la suite, nous nous pencherons sur le test de *boxplot*, les autres tests ayant des hypothèses d'application incompatibles avec les données que nous allons traiter.

Le test de *boxplot* (en français, boîte à moustaches) est l'un des plus utilisés. Il s'appuie sur le profil de la distribution d'une variable. Les quantiles permettent de délimiter des zones de valeurs aberrantes et des zones de valeurs rationnelles. Notons $\mathcal{Q}_i$ le $i$ème quartile des données et IQR la valeur interquartile, $IQR = \mathcal{Q}_3 - \mathcal{Q}_1$. Un intervalle, $[\mathcal{Q}_1; \mathcal{Q}_3]$, identifie les valeurs rationnelles : la *boîte*. Deux intervalles, $[\mathcal{Q}_1 - 1,5 \cdot IQR; \mathcal{Q}_1]$ et $[\mathcal{Q}_3; \mathcal{Q}_3 + 1,5 \cdot IQR]$, identifient les valeurs peu probables mais rationnelles : les *moustaches* de la boîte. Quant au complémentaire de ces intervalles, il identifie les valeurs aberrantes.

La figure 2.8 représente le *boxplot* de l'ensemble précédent : $\{10, 10, 12, 7, 8, 8, 9, 11, 6, 7, 42\}$. Pour cet exemple, les valeurs de $\mathcal{Q}_1$, $\mathcal{Q}_3$ et IQR valent respectivement $7,5$, $10,5$ et 3. La valeur 42 est clairement identifiée comme une valeur aberrante. Pour une distribution normale, ce test permet de rejeter au plus $0,7\%$ des données. Il est formalisé comme suit :

$$x \text{ est une valeur aberrante} \Leftrightarrow x \notin [\mathcal{Q}_1 - 1,5 \cdot IQR; \mathcal{Q}_3 + 1,5 \cdot IQR] \tag{2.10}$$
$$Q_i = i^{\text{ème}} \text{ quartile} \quad ; \quad IQR = Q_3 - Q_1$$

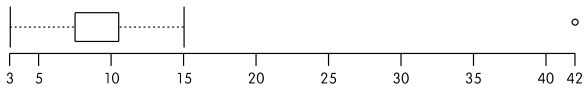


FIG. 2.8 – Exemple de *boxplot*

Ce test est adapté pour des distributions symétriques de forme gaussienne : la partie droite d'une telle distribution est le reflet de sa partie gauche. Mais il n'est pas adapté aux distributions asymétriques. Ces dernières possèdent une partie plus étalée que l'autre, comme présenté sur la figure 2.9. Si la partie plus étalée se situe à droite (respectivement à gauche), alors la distribution est asymétrique positive (respectivement négative).

Les travaux réalisés dans [VH04, HV06] proposent une modification du test *boxplot* afin de prendre en compte la caractéristique asymétrique des distributions. Ce test

se base sur une mesure de l'asymétrie d'une distribution, présentée dans [BHS04]. Il produit aussi un *boxplot*, avec des moustaches de taille fonction de la mesure d'asymétrie. La moustache de droite est plus grande (respectivement petite) que celle de gauche si la mesure reflète une asymétrie positive (respectivement négative). Il permet d'être plus souple que le précédent test pour les valeurs extrêmes. Son but est de ne pas considérer comme aberrantes des valeurs rationnelles dans les distributions asymétriques.

La mesure d'asymétrie notée $MC(D)$, pour un ensemble de données D, prend ses valeurs dans l'intervalle $[-1; 1]$. Une mesure de 0 indique que la distribution est symétrique. Une mesure positive (respectivement négative) indique que la distribution est asymétrique positive (respectivement négative). Elle se calcule de la manière suivante :

$$MC(D) = \underset{\substack{(x_i,x_j)\in D^2 \\ x_i<\mathcal{Q}_2<x_j}}{\text{median}}\, h(x_i, x_j) \quad \text{avec} \tag{2.11}$$
$$h(x_i, x_j) = \frac{(x_j - \mathcal{Q}_2) - (\mathcal{Q}_2 - x_i)}{x_j - x_i}$$

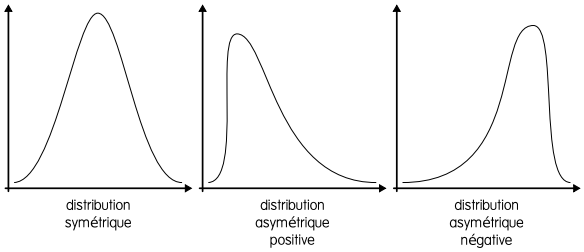


Fig. 2.9 – Exemples de distributions

Le test des valeurs aberrantes dépend de la mesure $MC(D)$. Il ressemble au test de *boxplot*, en se basant sur les valeurs des quartiles. Il se formalise de la manière suivante :

$$\text{Si } MC(D) \geq 0, \text{ et} \tag{2.12}$$
$$x \notin [\mathcal{Q}_1 - 1,5 \cdot e^{-4 \cdot MC(D)} \cdot IQR; \mathcal{Q}_3 + 1,5 \cdot e^{3 \cdot MC(D)} \cdot IQR]$$
$$\text{alors } x \text{ est une valeur aberrante}$$
$$\text{Si } MC(D) < 0, \text{ et} \tag{2.13}$$
$$x \notin [\mathcal{Q}_1 - 1,5 \cdot e^{-3 \cdot MC(D)} \cdot IQR; \mathcal{Q}_3 + 1,5 \cdot e^{4 \cdot MC(D)} \cdot IQR]$$
$$\text{alors } x \text{ est une valeur aberrante}$$

2.4.2 Identification des périodes de silence suspectes

Nous considérons comme hypothèse de base que les périodes de silence suspectes des environnements de pot de miel, qui risquent de biaiser les analyses statistiques,

vont se manifester comme des valeurs aberrantes, en considérant comme données de référence les intervalles de temps entre sessions observées sur le pot de miel. Dans ce cas, nous pouvons identifier ces périodes en appliquant le test de *boxplot* présenté dans la section 2.4.1.

Le tableau 2.3 montre, pour chaque environnement, les caractéristiques des variables aléatoires associées ($T'_{k,n}$), notées T'_k. L'intégralité de la période d'observation a été considérée (les 4 ans de collecte). La première colonne identifie l'environnement. La seconde colonne indique le nombre de réalisations de la variable aléatoire correspondante. La troisième colonne indique la valeur minimum. Les trois colonnes suivantes contiennent les quartiles. La septième colonne contient la valeur maximale de la variable. Quant aux deux dernières colonnes, elles contiennent la moyenne et l'écart type. Les unités sont indiquées entre parenthèses : (sec.) pour secondes, (min.) pour minutes et (jours) pour jours.

Tab. 2.3: Profils statistiques des environnements

env. k	$\lvert T'_k \rvert$	$\min(T'_k)$ (sec.)	$\mathcal{Q}_1(T'_k)$ (sec.)	$\mathcal{Q}_2(T'_k)$ (min.)	$\mathcal{Q}_3(T'_k)$ (min.)	$\max(T'_k)$ (jours)	$\overline{T'_k}$ (min.)	$\sigma(T'_k)$ (min.)
env. 1	61304	0	102	6	15	111	18	747
env. 2	27652	0	140	9	22	320	40	2792
env. 4	59934	0	55	8	23	75	20	475
env. 5	75897	0	43	2	6	29	5	160
env. 6	504651	0	29	1	2	110	3	302
env. 8	95564	0	101	5	13	76	14	425
env. 9	220758	0	58	3	9	17	7	74
env. 10	54414	0	29	1	2	36	3	234
env. 11	65464	0	200	9	25	27	24	161
env. 13	45658	0	1	12	45	57	36	396
env. 14	83777	0	148	9	24	54	20	292
env. 20	314380	0	32	1	2	15	2	56
env. 21	171035	0	93	4	11	64	9	229
env. 22	17603	0	0	1	21	9	20	112
env. 23	71090	0	53	3	9	10	7	66
env. 24	11394	0	121	5	11	169	41	2409
env. 25	167607	0	82	3	9	49	8	200
env. 26	79517	0	64	5	13	245	18	1361
env. 27	194437	0	80	4	9	14	7	49
env. 28	13247	0	605	37	99	132	104	1741
env. 30	21994	0	425	21	55	154	64	1503
env. 31	182300	0	49	3	8	36	7	152
env. 32	100656	0	137	6	14	67	13	311
env. 34	89320	0	80	6	17	41	14	208
env. 35	26616	0	286	17	42	59	43	648
env. 36	9088	0	82	3	7	0	5	5
env. 37	672	0	230	19	332	160	1039	9202

Suite page suivante

TAB. 2.3 – *Suite de la page précédente*

env. k	$\lvert T_k^{'} \rvert$	$\min(T_k^{'})$ (sec.)	$\mathcal{Q}_1(T_k^{'})$ (sec.)	$\mathcal{Q}_2(T_k^{'})$ (min.)	$\mathcal{Q}_3(T_k^{'})$ (min.)	$\max(T_k^{'})$ (jours)	$\overline{T_k^{'}}$ (min.)	$\sigma(T_k^{'})$ (min.)
env. 38	1844	0	5	0	1	0	4	16
env. 39	15366	0	154	7	19	9	17	135
env. 40	28132	0	37	1	3	29	4	250
env. 41	65846	0	161	7	17	12	16	74
env. 42	40802	0	105	7	28	12	27	136
env. 43	12325	0	156	7	15	96	22	1245
env. 44	41152	0	0	12	23	12	25	106
env. 45	77573	0	107	5	12	17	12	126
env. 47	11102	0	388	27	73	120	79	1644
env. 48	63066	0	129	5	12	41	12	262
env. 49	83007	0	143	5	14	6	11	58
env. 50	221180	0	30	1	4	9	3	34
env. 51	17738	0	16	5	12	86	18	942
env. 52	6029	0	212	14	48	39	124	1170
env. 53	72901	0	117	5	11	10	10	99
env. 54	86492	0	76	3	8	21	10	202
env. 55	52767	0	89	3	8	7	7	51
env. 56	105237	0	77	3	7	3	5	20
env. 57	268922	0	0	0	3	1	2	7
env. 58	10416	0	357	16	37	2	30	53
env. 59	60993	0	132	5	12	20	10	124
env. 60	38387	0	86	4	11	4	16	85
env. 61	47064	0	173	8	18	74	16	492
env. 62	26099	0	441	19	41	4	31	60
env. 63	47693	0	167	7	18	1	14	21
env. 64	72985	0	81	3	9	6	7	45
env. 65	41562	0	78	3	7	3	6	34
env. 66	10794	0	41	2	5	11	6	169
env. 67	12302	0	113	7	31	29	43	396
env. 68	15149	0	62	4	19	14	17	176
env. 69	21518	0	196	9	20	55	22	569
env. 70	4765	0	32	2	5	1	9	86
env. 71	58325	0	49	2	4	13	4	92
env. 72	21177	0	66	3	13	47	21	474
env. 73	17603	0	63	3	6	0	4	5
env. 74	26907	0	32	1	4	27	13	254
env. 75	30074	0	0	3	15	0	12	21
env. 76	20320	0	57	3	11	29	17	295
env. 77	80565	0	70	2	6	2	4	17
env. 78	17378	0	182	10	24	2	17	35
env. 79	50998	0	69	3	6	36	13	438

Suite page suivante

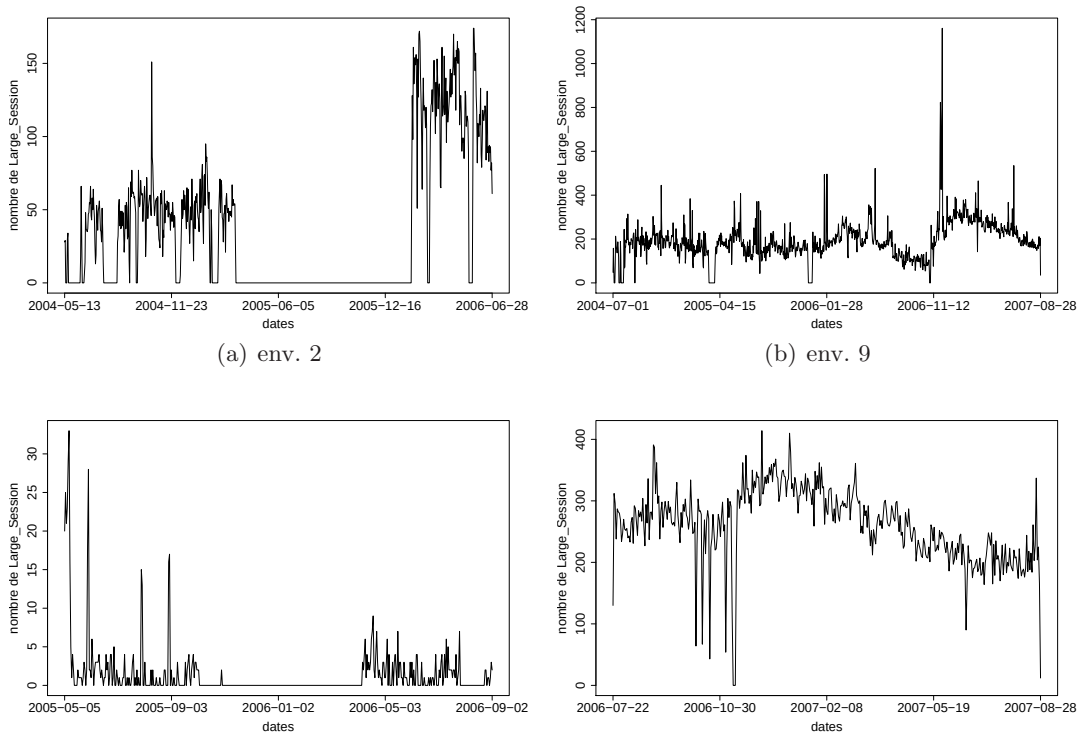
TAB. 2.3 – *Suite de la page précédente*

env. k	$\lvert T_k^{'} \rvert$	$\min(T_k^{'})$ (sec.)	$\mathcal{Q}_1(T_k^{'})$ (sec.)	$\mathcal{Q}_2(T_k^{'})$ (min.)	$\mathcal{Q}_3(T_k^{'})$ (min.)	$\max(T_k^{'})$ (jours)	$\overline{T_k^{'}}$ (min.)	$\sigma(T_k^{'})$ (min.)
env. 80	27369	0	150	6	13	3	10	30
env. 81	7930	0	202	10	34	23	34	411
env. 82	15662	0	203	8	17	7	15	134
env. 83	14569	0	97	4	9	1	6	14
env. 84	19284	0	165	6	14	1	10	15
env. 85	3028	0	584	30	71	2	58	112
env. 87	14436	0	84	3	7	1	5	18
env. 88	616	0	822	40	83	0	65	75

Nous constatons que les médianes sont nettement plus faibles que les moyennes. Autrement dit, largement plus de 50% des données possèdent des valeurs inférieures aux moyennes correspondantes. Les données sont généralement tassées sur un court intervalle proche de 0. Par contre, étant données les valeurs élevées des maxima, le nombre de données étalées sur un grand intervalle est non négligeable. Les densités de probabilité pour chaque environnement sont donc du type asymétrique positive. De plus, les quartiles sont différents d'un environnement à l'autre. La raison pourrait venir de la localisation géographique, du fait que certains environnements peuvent être la cible de plus d'attaques selon qu'ils se situent sur des réseaux haut débit ou leur nom de domaine. Pour certains environnements, la valeur du maximum est de l'ordre du mois. Pendant les périodes correspondantes, aucune session n'a été observée sur ces environnements. Ces valeurs élevées sont douteuses. Il est fort probable qu'elles soient des valeurs aberrantes, dues à des périodes de silence suspectes.

A titre d'illustration, la figure 2.10 présente l'évolution du nombre de sessions par jour pour quatre environnements : 2, 9, 37 et 56. En abscisse est reporté le temps et en ordonnées est reporté le nombre de sessions par jour. Un pic sur la courbe correspond à un jour pendant lequel l'environnement correspondant a été énormément ciblé par les attaquants. Par contre, un palier en 0 correpond à des jours pendant lesquels aucune session n'a été observée. Les environnements 9 et 56 présentent très peu de paliers. Leurs activités sont très élevées. Par contre, les environnements 2 et 37 présentent des paliers très importants. Prenons l'environnement 2 en exemple. Avant le 18 mars 2005, l'activité sur cet environnement était irrégulière mais présente. Juste après cette date et du jour au lendemain, l'activité a chuté subitement, se traduisant par un palier en 0 d'une longueur de 320 jours. 320 jours plus tard, l'activité repart à nouveau subitement. Clairement, nous sommes en présence d'un environnement qui a été déconnecté d'Internet. Dans un grand intervalle la possibilité de ne pas avoir observé une session est importante. Inversement, dans un petit intervalle cette possibilité est fortement réduite. Autrement dit, autant lorsque nous observons une session nous sommes sûrs de la disponibilité de l'environnement, autant lorsque le calme dure depuis longtemps nous sommes sûrs de l'indisponibilité.

L'identification des valeurs aberrantes est nécessaire. Elle doit donc être menée au cas par cas, en considérant les environnements indépendamment les uns des autres. Le



(a) env. 2 (b) env. 9

(c) env. 37 (d) env. 56

FIG. 2.10 – Evolution du nombre de sessions par jour pour certains environnements

type asymétrique positif des environnements nous amène à utiliser la version modifiée de la méthode *boxplot*.

Les résultats sont présentés dans le tableau 2.4. La deuxième colonne contient les valeurs de $MC(T'_{k,n})$. La troisième colonne identifie la valeur maximum permettant d'écarter les valeurs aberrantes, calculées en fonction de $MC(T'_{k,n})$. Plus exactement, la limite supérieure permet de considérer comme aberrantes toutes les valeurs qui lui sont supérieures. Toutes les valeurs minimums sont négatives et n'apparaissent donc pas dans ce tableau. Pour chaque environnement, toute valeur extérieure à cet intervalle peut être considérée comme aberrante. La dernière colonne représente le pourcentage de valeurs à considérer comme aberrantes. Les valeurs de la mesure $MC(T'_{k,n})$ sont positives. Elles permettent de bien identifier le caractère asymétrique positif des densités de probabilité. Les limites inférieures sont négatives. Or, les intervalles entre sessions sont positifs. Par conséquent, aucun petit intervalle n'est à considérer comme aberrant. Cette remarque est cohérente : un petit intervalle nous assure la disponibilité de l'environnement. Le pourcentage de valeurs considérées comme aberrantes est très faible. Nous réduisons ainsi fortement le risque de rejet à tort d'un grand intervalle.

Nous avons pu valider cette hypothèse pour un des environnements qui faisait l'objet d'une surveillance continue et pour lequel nous disposons d'informations en temps réel concernant sa disponibilité opérationnelle. Les résultats issus des tests des valeurs aberrantes sont conformes avec les données issues de l'observation opération-

nelle. Connaissant ces périodes, nous pouvons approximer la fonction $Up_k(t)$ (définie dans la section 2.2.2). Rappelons que cette fonction prend pour valeur 0 lorsque t se situe pendant une période de silence suspecte de l'environnement k et 1 sinon. Ainsi, lorsque t se situe dans un intervalle identifié comme aberrant, nous pouvons supposer que cette fonction prend la valeur 0. Cette fonction est utile pour sélectionner les données à analyser. Son approximation, notée $\widetilde{Up}_k(t)$, peut donc être définie comme suit (cf. équation 2.12) :

$$\widetilde{Up}_k(t) = \begin{cases} 0, & \text{si } \exists i / \lfloor \tau'_{k,i-1} \rfloor < t < \lfloor \tau'_{k,i} \rfloor \wedge T'_{k,i-1} > Q_3 + 1,5 \cdot \exp^{3 \cdot MC(T'_{k,n})} \cdot IQR \\ 1, & \text{sinon} \end{cases} \tag{2.14}$$

2.4.3 Sélection des données

Nous avons identifié les valeurs aberrantes et caractérisé les périodes de silence suspectes des environnements. A présent, nous devons identifier une grande sous-période durant laquelle un nombre donné d'environnements, n_e, sont actifs. Plus le nombre d'environnements choisis est élevé, plus les analyses comparatives sont significatives. Nous dirons d'un environnement qu'il est actif pendant une sous-période donnée si sa disponibilité est supérieure à un seuil fixé, s. Le critère associé est noté $Actif(k, a, b, s)$. Il prend ses valeurs dans l'ensemble $\{0, 1\}$. La valeur 1 indique que l'environnement k est actif durant la sous-période $[a, b]$ et la valeur 0 indique qu'il n'est pas actif.

env	$MC(T_k^{'})$	limite supérieure	% valeurs aberrantes
1	0,44	5551,97	1,12
2	0,4	7429,61	1,07
4	0,47	9880,12	0,16
5	0,4	2136,45	0,53
6	0,4	886,34	1,12
8	0,48	5393,33	0,61
9	0,41	3242,57	0,44
10	0,4	874,44	0,73
11	0,47	9602,77	1,79
13	0,53	23035,83	0,19
14	0,47	9547,52	0,36
20	0,39	883,42	1,11
21	0,39	3549,93	0,71
22	0,85	26318,02	0,2
23	0,41	3360,3	0,32
24	0,42	3842,19	1
25	0,43	3149,14	0,98
26	0,42	4727,75	0,8
27	0,37	2885,56	0,36
28	0,47	39080,25	0,6
30	0,49	22234,01	2,44
31	0,43	3133,42	0,57
32	0,4	4562,01	0,68
34	0,4	5654,07	0,48
35	0,42	14750,18	1,11
36	0,4	2145,11	0,37

env	$MC(T_k^{'})$	limite supérieure	% valeurs aberrantes
37	0,97	565928,6	0,74
38	0,65	733,18	8,46
39	0,51	8336,39	0,41
40	0,39	1199,06	0,8
41	0,45	6070,32	2,14
42	0,62	16775,46	0,55
43	0,37	4550,17	0,43
44	0,23	5687,49	5,93
45	0,46	4672,46	1,25
47	0,49	30156,38	1,05
48	0,39	3896,19	1,21
49	0,51	5840,64	0,2
50	0,42	1573,32	0,21
51	0,39	4420,22	0,7
52	0,59	26175,58	1,77
53	0,4	3512,96	1,56
54	0,39	2588,56	0,63
55	0,4	2696,36	0,92
56	0,35	2008,05	0,68
57	0,66	2207,87	0,04
58	0,43	12699,39	0,47
59	0,38	3522,83	1,02
60	0,44	4231,96	1,47
61	0,41	6089,22	0,52
62	0,37	12041,01	0,4
63	0,41	5839,24	0,93

env	$MC(T_k^{'})$	limite supérieure	% valeurs aberrantes
64	0,47	3522,63	0,65
65	0,43	2516,98	1,05
66	0,37	1517,52	0,48
67	0,71	24514,21	1,32
68	0,68	14050,11	0,04
69	0,38	6177,82	0,76
70	0,38	1597,45	0,82
71	0,37	1201,02	0,68
72	0,67	9033,37	1,56
73	0,36	1877,42	0,3
74	0,58	2403,43	7,13
75	0,58	8852,57	0,29
76	0,67	8065,19	1,98
77	0,33	1547,53	0,38
78	0,39	7571,92	0,4
79	0,38	1926,19	0,91
80	0,36	3666,56	0,49
81	0,58	18153,14	0,29
82	0,36	4949,25	0,41
83	0,35	2602,66	0,28
84	0,36	3985,87	0,48
85	0,4	22900,08	1,95
87	0,37	2139,73	0,41
88	0,35	22954,75	0,49

TAB. 2.4 – Résultat du test de *boxplot* modifié

$$Actif(k,a,b,s) = \begin{cases} 1, & \text{si } \frac{1}{b-a} \cdot \int_a^b \widetilde{Up}_k(t)dt > s \\ 0, & \text{sinon} \end{cases} \tag{2.15}$$

L'identification de la plus grande période pour laquelle au moins n_e environnements sont actifs se fait de la manière suivante. Nous considérons, au départ, une sous-période égale à la durée totale des observations. Pour cette sous-période, nous calculons le nombre d'environnements actifs. Si ce nombre est supérieur ou égal à n_e, cette sous-période nous satisfait et nous stoppons la recherche. Si ce nombre est inférieur à n_e, nous considérons une sous-période plus petite de 1 heure ainsi que toutes les possibilités d'intercaler cette sous-période dans la durée totale d'observation. Nous réitérons pour chacune de ces possibilités. Le parcours se fait donc de manière exhaustive et ordonnée. En débutant l'algorithme par la plus grande sous-période et en réduisant cette période par pas de 1 heure, nous assurons d'obtenir la plus grande sous-période satisfaisant nos critères. L'algorithme est présenté à la figure 1.

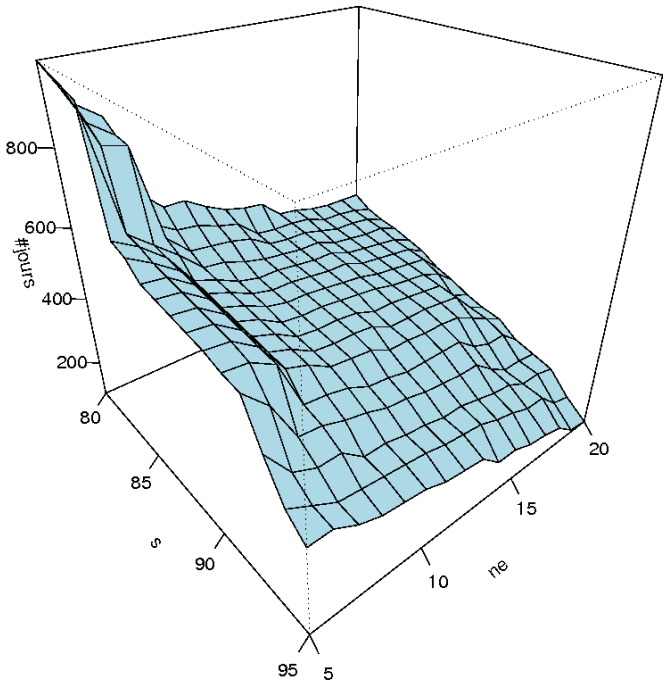
Algorithme 1 Algorithme de sélection de la sous-période

Entrées: le pas de recherche $pas \in \mathbb{R}^+$
Entrées: la date de début des observations $debut \in \mathbb{R}^+$
Entrées: la date de fin des observations $fin \in \mathbb{R}^+$
Entrées: le seuil de disponibilité s
Entrées: le nombre d'environnements choisis n_e
Sorties: l'ensemble des environnements sélectionnés l
Sorties: la date de début de la sous-période a
Sorties: la date de fin de la sous-période b

```
nb_pas ← (fin−debut)/pas
trouve ← faux
longueur_sous_periode ← nb_pas
tant que longueur_sous_periode ≥ 1 ∧ ¬trouve faire
  decalage ← nb_pas − longueur_sous_periode + 1
  tant que decalage ≥ 1 ∧ ¬trouve faire
    a ← debut + decalage ∗ pas
    b ← a + longueur_sous_periode ∗ pas
    l ← {k ∈ envs/Actif(k, a, b, s) = 1}
    si |l| ≥ n_e alors
      trouve ← vrai
    finsi
    decalage ← decalage − 1
  fin tant que
  longueur_sous_periode ← longueur_sous_periode − 1
fin tant que
```

Nous avons appliqué cet algorithme à nos données en considérant différentes valeurs pour n_e et s. Les résultats sont présentés à la figure 2.11. La première représentation est sous forme d'un graphique en trois dimensions. L'ordonnée et l'abscisse du plan horizontal correspondent repectivement au nombre d'environnements choisi et au seuil de disponibilité. Pour des valeurs fixées pour ces deux critères, la longueur de la plus

longue sous-période les satisfaisant est reportée en hauteur, en jours. La seconde représentation reprend quelques unes des valeurs intéressantes.



(a) Sous forme graphique

s	n_e	$\#jours$
80%	5	994
85%	5	651
90%	5	588
95%	5	343
80%	8	637
85%	8	490
90%	8	455
95%	8	287
80%	10	581
85%	10	441
90%	10	399
95%	10	273
80%	15	448
85%	15	413
90%	15	350
95%	15	189
80%	20	420
85%	20	343
90%	20	259
95%	20	98

(b) Sous forme tabulaire

FIG. 2.11 – Evolution du nombre de jours pour la sous-période, en fonction du nombre de périodes de silence suspectes et du nombre d'environnements minimum

Il est à noter que si nous augmentons le nombre d'environnements souhaité, la période satisfaisant le seuil de disponibilité devient plus petite. Le même constat peut être effectué par rapport au seuil de disponibilité fixé. Plus il est élevé, plus la période satisfaisant le critère est courte. Un compromis est donc nécessaire pour obtenir une période suffisamment longue avec un nombre d'environnements suffisamment élevé. Nous avons choisi les valeurs $n_e = 8$ et $s = 80\%$. Le nombre d'environnements choisi est suffisant pour réaliser des analyses comparatives significatives, tout en permettant facilement de présenter les résultats.

En appliquant l'algorithme précédent, nous avons sélectionné une sous-période de 637 jours, du 27 octobre 2005 au 26 juillet 2007 et 8 environnements : 9, 13, 14, 28, 31, 32, 42 et 62. Ces 8 environnements sont répartis entre la France, la Belgique, la Pologne, l'Italie, l'Allemagne et le Royaume-Uni. Nous assurons, pour la sous-période, au moins une disponibilité supérieure à 80% pour chacun des environnements. De plus, la sous-période considérée est assez longue pour mener des analyses significatives (environ deux ans). Ces disponibilités et les nombres d'intervalles à considérer pour la suite des analyses sont présentés dans le tableau 2.5. Le minimum n'est pas présenté dans ce tableau, car il égale 0 pour les 8 environnements.

Dans cette section, nous avons identifié les valeurs aberrantes pour chacun des environnements. Cette identification permet de caractériser la disponibilité. Ensuite,

env. k	$\lvert T'_{k,n}\rvert$	$\mathcal{Q}_1(T'_k)$ (sec.)	$\mathcal{Q}_2(T'_k)$ (min.)	$\mathcal{Q}_3(T'_k)$ (min.)	$\max(T'_k)$ (min.)	$\overline{T'_k}$ (min.)	$\sigma(T'_k)$ (min.)	disp. (%)
9	134161	56	3	8	53	6	7	93
13	15742	538	32	73	382	52	59	89
14	42670	107	7	24	158	18	25	85
28	10200	578	35	96	650	73	100	82
31	90580	76	4	11	52	8	9	81
32	65962	161	7	16	76	11	12	84
42	38826	102	7	25	278	19	29	84
62	25042	435	19	40	199	29	30	80

TAB. 2.5 – Caractéristiques des environnements séléctionnés pour la sous période

nous avons déterminé le sous ensemble des environnements et une sous période tels que les disponibilités des environnements du sous ensemble, pendant la sous période, sont supérieures à 80%. Nous minimisons ainsi les risques d'avoir des résultats biaisés dûs à des indisponibilités des environnements. Les analyses qui suivent se basent sur ces données.

2.5 Modélisation des intervalles entre attaques

L'identification d'un modèle reflétant les temps entre sessions est utile pour caractériser le processus d'occurrence des attaques. Il peut servir pour guider les administrateurs en servant d'indicateur. De tels modèles peuvent aussi servir à générer des traces d'attaques représentant du trafic malveillant réel, pouvant être utilisé pour des test de validation de systèmes et mécanismes de protection vis-à-vis des malveillances.

Différents modèles et distributions de probabilité peuvent être testés pour identifier ceux qui correspondent mieux aux données observées. Dans le cadre de nos travaux, nous avons étudié plus particulièrement des modèles paramétriques où les valeurs des paramètres doivent être estimées à partir des données observées.

Nous distinguons deux étapes principales. Tout d'abord, une étape d'inférence permet de calibrer le modèle. L'objectif visé lors de cette étape est de déterminer, sur la base des observations, les paramètres du modèle qui permettent de minimiser l'écart entre les valeurs observées et les valeurs estimées par le modèle. Pour ce faire, l'écart entre le modèle et les observations est déterminé en fonction des paramètres du modèle. La fonction obtenue est la fonction de coût. Des algorithmes d'optimisation font alors évoluer ces paramètres jusqu'à trouver les paramètres pour lesquels la fonction de coût est minimisée. Ensuite, une étape de validation permet de vérifier l'adéquation des valeurs estimées par le modèle aux valeurs observées. Cette étape se base sur des tests statistiques pour aider à la décision de rejet ou d'acceptation du modèle.

Dans la suite, nous présentons dans un premier temps les algorithmes d'optimisation permettant d'estimer les paramètres des modèles. Ensuite, nous présentons les tests statistiques utilisés pour estimer l'ajustement d'un modèle. Pour finir, nous appliquons ces outils aux données.

2.5.1 Estimation des paramètres : l'algorithme EM

La vraisemblance quantifie la probabilité qu'un échantillon observé $\underline{x} = (x_i)_{1 \leq i \leq n}$ soit issu d'un échantillon théorique $\underline{X} = (X_i)_{1 \leq i \leq n}$, lui-même issu d'une loi théorique $f(x|\theta)$. La probabilité que l'échantillon observé soit une réalisation de l'échantillon théorique est égale au produit des probabilités que chaque donnée de l'échantillon observé x_i soit une réalisation de la variable aléatoire de l'échantillon théorique correspondante X_i. Cette quantité est notée $L(\underline{x}, \theta)$.

$$L(\underline{x}, \theta) = \prod_{i=1}^{n} f(x_i|\theta) \tag{2.16}$$

La méthode du maximum de vraisemblance permet d'inférer les paramètres d'une loi d'un échantillon. Elle part de l'hypothèse que l'échantillon observé est effectivement issu de l'échantillon théorique. Son principe est le suivant : nous avons observé l'échantillon $\underline{x}$; cet échantillon est une réalisation de l'échantillon théorique $\underline{X}$; la probabilité que l'échantillon observé soit effectivement observé doit donc être très élevée ; la vraisemblance doit, elle aussi, être très élevée. La méthode du maximum de vraisemblance infère les paramètres θ qui maximisent la vraisemblance.

La vraisemblance est un produit. Maximiser un produit revient à trouver les valeurs qui annulent sa dérivée. Cette étape peut être fastidieuse pour une taille d'échantillon observé très élevée. Le logarithme – fonction croissante sur $\mathbb{R}$ utilisée, entre autres, pour réaliser des bijections – est donc employé pour substituer une somme à ce produit et ainsi simplifier le traitement. Maximiser la vraisemblance revient à maximiser le logarithme de la vraisemblance. D'un point de vue pratique, différentes procédures d'optimisation numérique existent pour trouver le maximum de vraisemblance. Citons, par exemple, la procédure itérative de Newton-Raphson, la procédure de quasi Newton et la procédure numérique de recherche directe d'optimum de Powell.

Parfois, la loi théorique manipulée possède une expression compliquée. Elle peut être un mélange de densités de probabilités. Un mélange de densité, notée $f(x|\Pi, \Theta)$, est une somme pondérée de densités de probabilité. Trouver les paramètres d'un mélange avec les méthodes classiques d'optimisation (Nelder-Mead, Newton, ...) est très difficile. Par la méthode du maximum de vraisemblance, le logarithme pour un mélange de densités nous amène à traiter un logarithme d'une somme, qui est difficile à maximiser. Le principal problème vient du fait que nous ne savons pas a priori quelles sont les données générées par chacune des parties du mélange.

$$\Theta = (\theta_i)_{1 \leq i \leq l} \tag{2.17}$$

$$\Pi = (\pi_i)_{1 \leq i \leq l} \qquad \pi_i > 0, \forall i \quad \text{et} \quad \sum_{i=1}^{l} \pi_i = 1 \tag{2.18}$$

$$f(x|\Pi, \Theta) = \sum_{i=1}^{l} \pi_i \cdot f_i(x|\theta_i) \tag{2.19}$$

L'algorithme Espérance-Maximisation (EM) permet de contourner cette difficulté. EM est un algorithme itératif très utilisé en statistique pour l'estimation de paramètres. Il a fait l'objet d'une importante littérature[DLR77, Bil97, Del02, Col97]. Il

est utilisé dans plusieurs domaines, de la construction d'une chaîne de Markov cachée à l'identification des paramètres d'un processus ON/OFF. Son application à l'estimation des paramètres d'un mélange de densités est très efficace. L'idée de base de cet algorithme est simple : alterner entre l'étape d'estimation de l'espérance de la vraisemblance (E) et l'étape de maximisation de cette vraisemblance (M). L'estimation des paramètres est améliorée à chaque itération.

Dans l'algorithme Em, les associations entre les données et les parties du mélange de densités sont considérées comme des valeurs manquantes. Nous observons $\underline{x}$ et il nous manque les associations notées $\underline{y}$. $y_{i,j}$ vaut 1 si la donnée x_i a été générée par la densité $f_j(x|\theta_i)$, sinon elle vaut 0. Connaissant une approximation de Θ, l'étape E permet de trouver les valeurs moyennes pour les données manquantes $\underline{y}$ correspondant aux associations. Connaissant a priori les valeurs des données manquantes, l'étape M permet de trouver une meilleure approximation de Θ. L'algorithme 2 présente une implémentation. Sachant que la phase M est fortement liée au mélange de densité employé, l'implémentation ne peut que très difficilement être générique. Il est plus facile de la réaliser au cas par cas.

$$\underline{y} = (y_{i,j})_{1 \leq i \leq n, 1 \leq j \leq l} \tag{2.20}$$

Algorithme 2 Algorithme Em

Entrées: $\underline{x}, f(x|\Pi, \Theta)$
Sorties: Π, Θ

```
tant que L'algorithme n'a pas convergé faire
  - - Etape E - Calcul des valeurs moyennes pour les données manquantes par la règle de
  - - Bayes
  - - - - - - - - - - - - - - - - - - - - - - - -
  i ← 1
  tant que i ≤ n faire
    j ← 1
    tant que j ≤ l faire
      y_{i,j} ← π_j · f_j(x_i|θ_j) / Σ_{k=1}^{l} π_k · f_k(x_i|θ_k)
      j ← j + 1
    fin tant que
    i ← i + 1
  fin tant que
  - - Etape M - Détermination d'un meilleur Θ
  - - - - - - - - - - - - - - - - - - - - - - - -
  Θ ← argmax_Θ Σ_{i=1}^{n} Σ_{j=1}^{l} y_{i,j} · log(π_j · f_j(x_i|θ_j))
  - - Récupération des pondérations optimales
  - - - - - - - - - - - - - - - - - - - - - - - -
  j ← 1
  tant que j ≤ l faire
    π_j ← (1/n) Σ_{i=1}^{n} y_{i,j}
    j ← j + 1
  fin tant que
fin tant que
```

2.5.2 Validation des modèles : tests statistiques

La validation des modèles a pour objectif de vérifier dans quelle mesure le modèle estimé est acceptable pour décrire les données observées. L'analyse de données peut être vue comme une série de questions permettant de guider le raisonnement. Il est important de savoir si la réponse hypothétique, que l'on apporte à une question, a un sens. Le but d'un test statistique est d'aider à déterminer si une hypothèse est acceptable. Pour ce faire, la plupart des tests fonctionnent sur la base d'une mesure de la distance entre la réalisation et la théorie, exprimée sous la forme d'une statistique. Pour chacun des tests, des études préalables ont permis de déterminer le comportement de cette statistique en fonction de la réalisation et de la théorie, aboutissant à la détermination de la distribution de probabilité de la statistique. Une statistique faible a une forte probabilité d'être associée à une bonne théorie et, inversement, une distance élevée a une forte probabilité d'être associée à une mauvaise théorie. De manière générale, plus cette statistique est élevée et moins l'hypothèse doit être envisagée. Des indicateurs ont été déterminés sur la base de cette statistique et de sa distribution, afin de connaître les risques encourus dans le choix d'acceptation ou rejet de l'hypothèse. Ces indicateurs se nomment *p-valeurs*. Une p-valeur prend ses valeurs dans l'intervalle $[0; 1]$. Nous admettons qu'une p-valeur supérieure à $0,05$ nous indique qu'il n'y a aucune raison de rejeter l'hypothèse, pour un seuil de signification allant jusqu'à 5%. Ce seuil correspond à la probabilité que le test rejette à tort l'hypothèse. Dans la suite, nous présentons le test du χ^2 et le test de Kolmogorov-Smirnov, tous deux répandus en statistique.

2.5.2.1 Test du χ^2

Le test du χ^2 est utilisé pour comparer une distribution expérimentale des données à une distribution théorique lorsque la variable aléatoire manipulée est discrète. Néanmoins, une adaptation au cas où la distribution est continue existe également. Les données sont réparties en n classes. La statistique de ce test, notée S, est la somme des écarts entre les densités de probabilité empirique, notée E_i et théorique, notée T_i. Elle suit une distribution du χ^2 de paramètre m. Ce paramètre se nomme le degré de liberté. Il égale le nombre de classes moins le nombre de paramètres du modèle estimés à partir des données. Une petite valeur pour la statistique indique une bonne adéquation entre l'expérience et la théorie. Pour déterminer la frontière entre le choix de rejet et d'acceptation, nous nous donnons a priori un risque de nous tromper, α, souvent égal à 5%. A partir du nombre de degrés de liberté, les tables du χ^2 nous donnent la distance critique, notée χ_0^2, qui a une probabilité de dépassement égale au risque. Quant à la p-valeur, elle représente la probabilité que la statistique soit inférieure à la distance critique. La démarche de ce test est présentée dans la figure 2.12.

2.5.2.2 Test de Kolmogorov-Smirnov

Le test de Kolmogorov-Smirnov est un test d'ajustement non paramétrique permettant de confronter des données à une loi de probabilité théorique. Dans les grandes lignes, il suit la même démarche que le test du χ^2. Il teste l'hypothèse selon laquelle ces données ont été engendrées par une loi de probabilité qui admet comme modèle

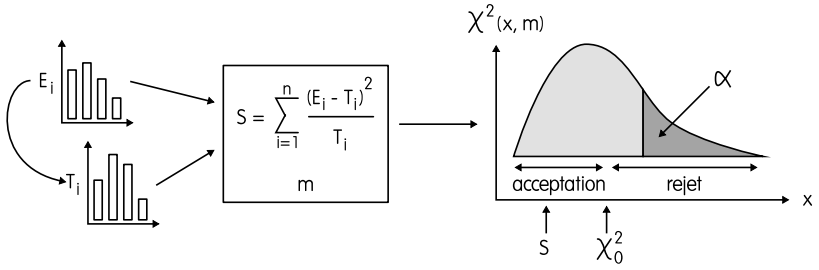


FIG. 2.12 – Test du χ^2

convenable la loi de probabilité théorique. Il est naturellement utilisé lorsque la loi théorique est une loi de probabilité continue. Son principe repose sur l'étude de la différence entre la fonction de répartition empirique des données et la fonction de répartition théorique. Ce test statistique fournit une statistique permettant de guider le choix de rejet ou d'acceptation de l'hypothèse. Elle correspond à l'écart maximum entre les deux fonctions de répartition. Parfois, le test de Kolmogorov-Smirnov est difficile à appliquer, de par la nature des données manipulées. Pour pallier ce problème, la technique d'échantillonnage est utilisée[SMQ93]. Plusieurs échantillons sont générés à partir des données et confrontés à la loi théorique par le test statistique, en l'occurrence le test de Kolmogorov-Smirnov. En fonction d'un seuil de confiance, le test réussit si la majorité des tests ne rejette pas l'hypothèse.

2.5.3 Application aux processus d'attaque observés

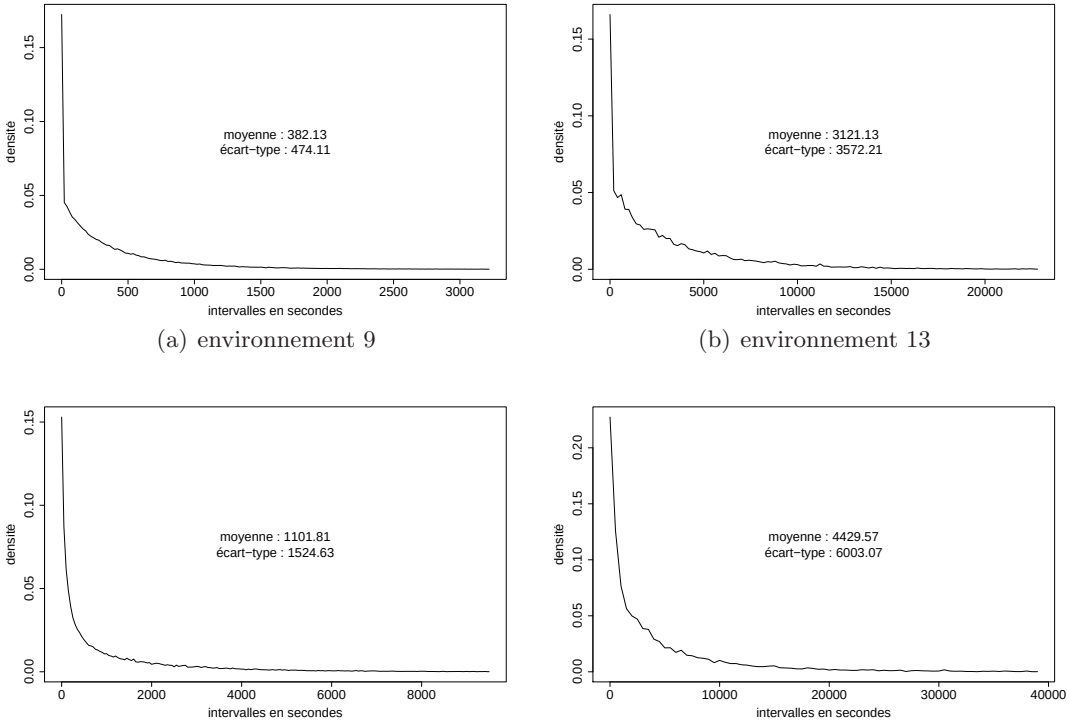
Les données que nous manipulons représentent les réalisations d'une variable aléatoire représentant le temps entre deux sessions consécutives. Cette variable aléatoire est continue. Or, nous disposons de réalisations arrondies à la seconde. Les données sont donc catégorisées et chaque catégorie couvre une seconde. Pour réaliser les tests statistiques, nous utiliserons donc trois tests. Le premier est le test du χ^2, adapté aux données catégorisées. Le second test est le test de Kolmogorov-Smirnov tel que présenté dans [SMQ93], utilisé lorsque l'ajustement à partir de modèle associé à une variable continue est difficile. Le troisième test est le coefficient de corrélation (cf. section 2.6.2), utilisé pour juger de la bonne corrélation entre la densité de probabilité empirique et théorique.

Dans un premier temps, nous avons observé les profils des différentes densités de probabilité. Ces observations ont permis de choisir plusieurs candidats. Ensuite, nous avons utilisé la méthode EM pour l'identification des paramètres optimaux pour chacun de ces candidats. Nous avons alors testé l'adéquation de ces modèles ainsi établis.

2.5.4 Densités de probabilité empiriques

La figure 2.13 présente les densités de probabilité empirique de quatre des 8 environnements : 9, 13, 14 et 28. Nous remarquons que la forme des courbes est décroissante et proche de celle d'une fonction exponentielle. La probabilité qu'un intervalle entre sessions soit faible est très élevée. D'un point de vue physique, cela correspond à

un nombre important d'attaques qui arrivent très proches les unes des autres. Ces remarques s'appliquent à tous les environnements. Etant donné la forme de ces courbes, nous avons identifié des distributions candidates pour former un modèle.



(a) environnement 9 (b) environnement 13

(c) environnement 14 (d) environnement 28

Fig. 2.13 – Densités de probabilité des environnements 9, 13, 14 et 28

2.5.5 Modélisation

Pour trouver un modèle satisfaisant, nous avons testé plusieurs distributions. Nous avons, avant tout, testé l'adéquation de la loi exponentielle aux données. Elle a été choisie pour trois raisons. Elle est simple. Elle correspond a priori à la forme des données. Elle est très largement utilisée dans le domaine des réseaux et de la fiabilité des systèmes en général. Beaucoup de travaux utilisent ce modèle pour représenter le temps qui s'écoule entre deux paquets réseaux consécutifs. Cependant, les tests statistiques ont rejeté cette distribution, pour tous les environnements. Il en est de même pour toutes les distributions de type exponentielle (à savoir Weibull, lognormale, ...). La principale raison provient du fait qu'elle est moins courbée que les densités de probabilité empiriques des environnements. En d'autres termes, elle ne permet pas de bien modéliser l'importance des arrivées en rafale des attaques. Cette constatation nous a amené à considérer une distribution permettant de modéliser le principe du $20-80$. Ce principe, utilisé en économétrie, affirme que 20% d'une population est responsable de 80% des évènements. Une distribution adaptée à ce principe est la distribution de Pareto généralisée. Elle nous permet a priori de modéliser le fait que « 80% des intervalles sont des petits intervalles ». Cependant, la confrontation de

cette distribution avec les données observées n'a pas été concluante. Elle reflète bien les intervalles petits mais ne représente pas correctement les grands intervalles, pour les données que nous avons analysées. A l'instar de la distribution exponentielle, les tests l'ont rejetée.

Les distributions de la famille des exponentielles et la distribution de Pareto généralisée permettent chacune de bien modéliser une partie des données. Les distributions de la famille des exponentielles s'ajustent bien, à vue d'oeil, à la queue de la densité de probabilité. Quant à la distribution de Pareto généralisée, elle s'ajuste bien au début de la densité de probabilité. Elles sont complémentaires. Aussi, nous avons construit un mélange entre ces deux distributions, dans l'optique d'obtenir une bonne adéquation en les utilisant conjointement. La densité associée est la suivante :

$$f(t) = \Pi \cdot \frac{1}{\sigma} \cdot (1 - \frac{\epsilon t}{\sigma})^{\frac{1}{\epsilon}-1} + (1 - \Pi) \cdot \frac{k}{\lambda} \cdot (\frac{t}{\lambda})^{k-1} \cdot e^{-(\frac{t}{\lambda})^k} \tag{2.21}$$

Le terme pondéré par la probabilité Π correspond à la distribution de Pareto généralisée, avec les paramètres ϵ et σ. Le terme pondéré par la probabilité complémentaire, $(1 - \Pi)$, correspond à la distribution de Weibull, de paramètres λ et k positifs ($k = 0$ correspond à une loi exponentielle).

De manière générale, ce mélange de distribution permet de modéliser deux comportements différents. La distribution de Pareto généralisée permet de modéliser les rafales d'attaques. Quant à la distribution de Weibull, elle permet de modéliser le trafic régulier. La combinaison de ces deux distributions permet d'aboutir à un modèle de mélange. L'importance d'un comportement est revélé par la valeur de Π. Une valeur proche de 1 indique que les rafales d'attaques sont très importantes, comparées au trafic malveillant régulier. Inversement, une valeur proche de 0 indique que les rafales d'attaques sont moins importantes, comparées au trafic malveillant régulier. Ce mélange de distributions contient 5 paramètres. Pour les estimer, nous utilisons l'algorithme Em (cf. section 2.5.1).

env. k	p-valeur mélange	p-valeur lognormale	p-valeur exponentielle	p-valeur weibull	p-valeur pareto
9	0,209	0	0	0	0
13	0,3923	0	0	0	0
14	0,8472	0	0	0	0
28	0,1855	0	0	0,0028	0
31	0,1001	0	0	0	0
32	0,1101	0	0	0	0
42	0,9424	0	0	0	0
62	0,611	0	0	0,0644	0

Tab. 2.6 – Résultat du test de χ^2 sur les données en considérant le mélange de distributions, la loi lognormale, la loi exponentielle, la loi de weibull et la loi de pareto

Les résultats sont présentés dans le tableau 2.6. La première colonne indique l'environnement testé. Les cinq colonnes suivantes fournissent les p-valeurs du test de χ^2 pour le mélange de distribution, la distribution lognormale, la distribution exponentielle, la distribution de Weibull et la distribution de Pareto généralisée, toutes

env. k	p-valeur du test de χ^2	test de Kolmogorov-Smirnov	coefficient de corrélation
9	0,209	✓	0,9999
13	0,3923	✓	0,9989
14	0,8472	✓	0,9998
28	0,1855	✓	0,9992
31	0,1001	✓	0,9998
32	0,1101	✓	0,9992
42	0,9424	✓	0,9999
62	0,611	✓	0,9991

TAB. 2.7 – Résultat du test de χ^2, de Komogorov-Smirnov et coefficient de corrélation pour le mélange de distributions

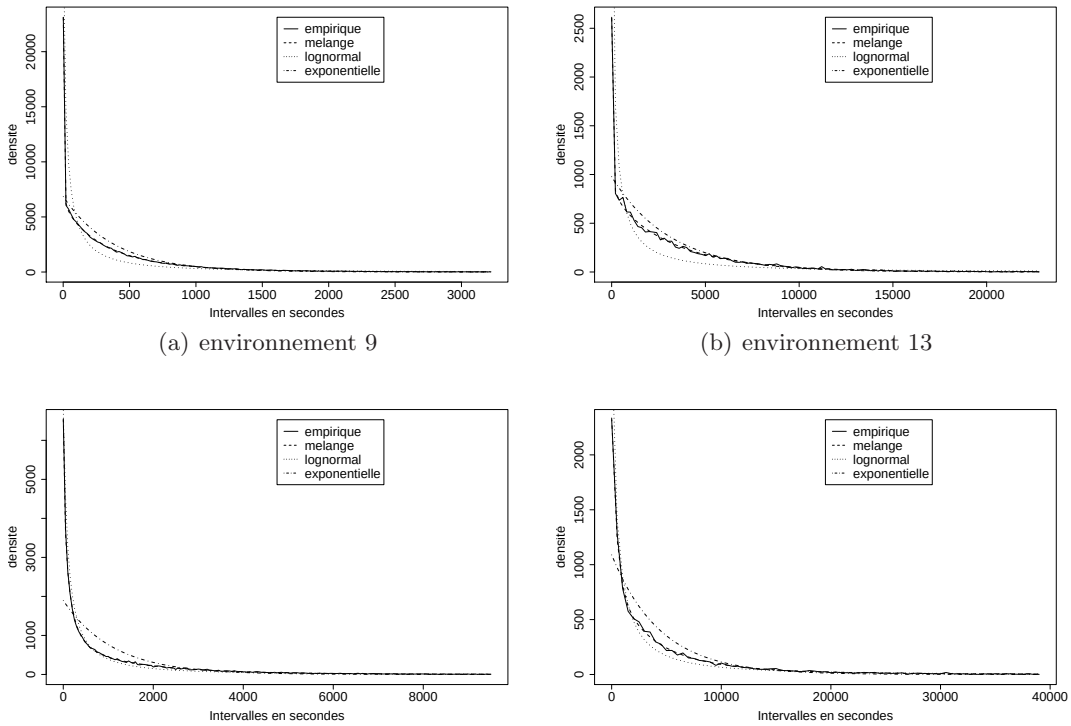
cinq ajustées. Le tableau 2.7 présente les tests du χ^2, de Kolmogorov-Smirnov et le coefficient de corrélation en considérant le mélange de distribution. La nature catégorisée des données se prêtant mal au test de Kolmogorov-Smirnov habituel, nous avons utilisé la version avec échantillonnage : une croix indique que le test est accepté.

env. k	Π	σ	ϵ	k	λ
9	0,8474	386,3455	0,1699	0,2129	1,0938
13	0,8277	3484,144	0,045	0,2137	105,119
14	0,6125	1616,785	0,1132	0,7751	151,3294
28	0,7906	4316,69	0,3457	1	445,6481
31	0,7802	522,7253	0,1313	0,1101	270,2345
32	0,9001	677,7271	0,1429	0,5078	197,2162
42	0,5718	1835,468	0,075	0,7941	156,073
62	0,8789	1856,019	0,038	0,6085	606,6787

TAB. 2.8 – Paramètres des modèles de mélange ajustés

Les p-valeurs sont très bonnes pour le mélange de distributions. Etant toutes supérieures à $0,10$, ce modèle est convenable en considérant des tests statistiques avec un niveau de signification jusqu'à 10%. Les coefficients de corrélation entre les valeurs estimées et les valeurs observées le confirment. Par contre, les autres modèles sont rejetés pour tous les environnements. Concernant la loi exponentielle, son rejet est en accord avec les travaux menés dans [PF94]. Dans ces travaux, l'auteur montre que le modèle exponentiel n'est pas adapté pour modéliser toutes les caractéristiques des trafics réseaux. Nous ne travaillons pas au niveau des paquets, mais à un niveau plus élevé et le modèle exponentiel semble également ne pas être adapté quand nous analysons le trafic malveillant. La figure 2.14 présente l'ajustement du modèle pour les environnements 9, 13, 14 et 28. La figure 2.15 reprend les ajustements des environnements 9 et 13 pour les tracer avec des échelles logarithmiques. Nous voyons clairement que la distribution de mélange se confond avec la distribution empirique.

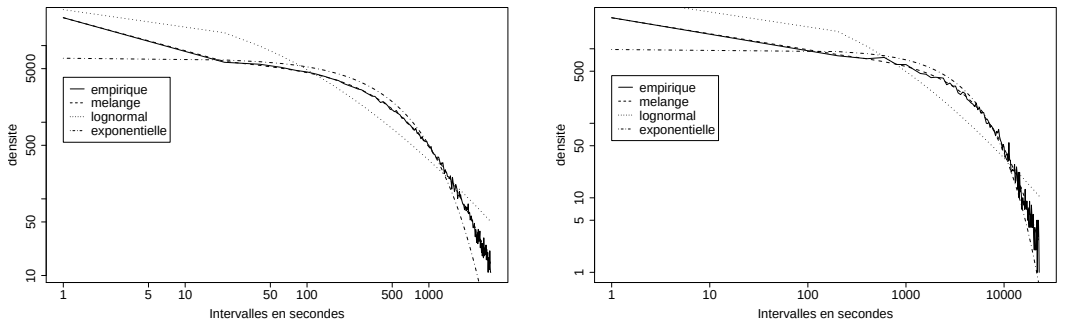
Nous pouvons constater que le modèle de mélange s'ajuste bien avec la majorité des environnements. Ce modèle peut être utilisé afin de générer des traces de trafic



(a) environnement 9 (b) environnement 13

(c) environnement 14 (d) environnement 28

FIG. 2.14 – Ajustement du modèle de mélange et comparaison avec d'autres modèles



(a) environnement 9 (b) environnement 13

FIG. 2.15 – Ajustement du modèle de mélange, échelles logarithmiques

malveillant. Cette orientation permet d'enrichir les outils de simulation de trafic réseau en prenant en compte les malveillances.

2.6 Analyse des corrélations

Les modèles présentés dans cette section concernent l'évolution du nombre de sessions observées sur les différents environnements et les corrélations possibles entre

les sessions originaires de différents pays. Dans un premier temps, nous présentons l'évolution dans le temps du nombre de sessions pour certains environnements. Les similitudes dans ces évolutions nous ont amené à considérer des outils permettant d'établir des modèles de régression et d'en juger l'ajustement. Ces outils sont présentés dans un second temps. Pour finir, nous appliquons ces outils sur les données représentant l'évolution du nombre de sessions dans le temps.

2.6.1 Evolution dans le temps du nombre de sessions

Les sessions sont réalisées par des adresses sur Internet. Les origines géographiques de ces adresses ont été récupérées. Il est alors possible d'identifier, pour chaque environnement, l'ensemble des sessions réalisées par des adresses appartenant à un pays en particulier. Le tableau 2.9 présente, pour chaque environnement, la répartition des sessions sur différents pays en pourcentage. Certains pays sont très actifs sur certains environnements. Par exemple, pour l'environnement 9, les sessions provenant des Etats-Unis forment 24% de l'activité de cet environnement. Il est intéressant de savoir si cette activité intense, réalisée par les Etats-Unis, est localisée dans le temps ou si elle est homogènement répartie. La figure 2.16 présente l'évolution du nombre de sessions par jour et toute origine confondue ainsi que l'évolution du nombre de sessions en provenance des Etats-Unis. La similarité entre les tendances des deux courbes est assez frappante. Certes, elles ne sont pas de même amplitude, mais les pics de l'un se retrouvent facilement chez l'autre. Ces deux courbes ont de fortes chances d'être corrélées. Les modèles de régression linéaire sont généralement utilisés pour analyser ce type de tendances. La suite présente les outils pour établir les modèles de régression linéaire.

env. k	$\|LS_i\|$	Chine	Canada	France	Etats-Unis	Pologne	Espagne
global	425630	17%	3%	3%	20%	2%	2%
9	134591	10%	2%	3%	24%	2%	1%
13	15813	37%	2%	2%	20%	1%	1%
14	42902	32%	5%	3%	19%	1%	2%
28	10261	21%	2%	2%	27%	16%	1%
31	91534	13%	2%	3%	11%	3%	3%
32	66490	12%	3%	5%	23%	3%	2%
42	38916	33%	7%	3%	20%	1%	1%
62	25123	12%	2%	3%	27%	2%	2%

TAB. 2.9 – Proportion des sessions par environnement en considérant la Chine, le Canada, la France, les Etats-Unis, la Pologne et l'Espagne

2.6.2 Régression linéaire et coefficient de corrélation

En statistiques, deux grandeurs $X = (x_i)_{1 \leq i \leq n}$ et $Y = (y_i)_{1 \leq i \leq n}$ peuvent être liées par une relation linéaire, $X = f(Y|\Theta) + \epsilon$, où X est la variable expliquée et Y la variable explicative. La fonction $f(Y|\Theta)$ est l'équation de régression de la variable X

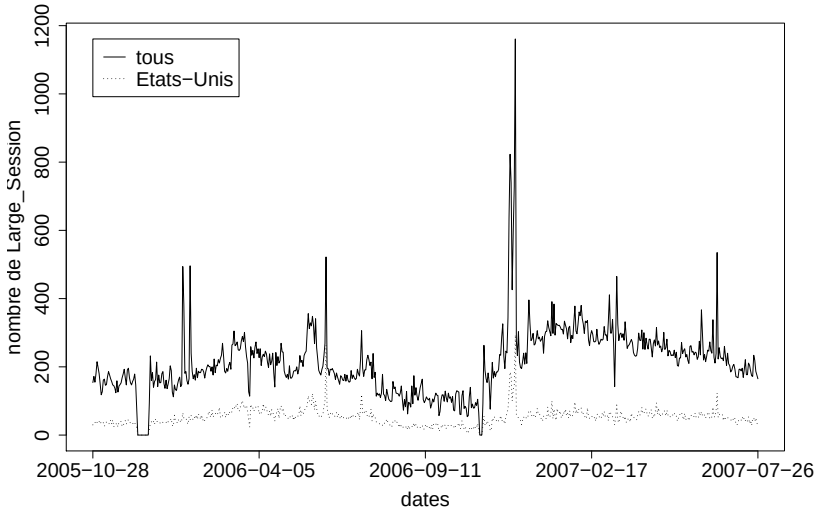


FIG. 2.16 – Evolution du nombre de sessions toutes origines confondues et évolution du nombre de sessions provenant des Etats-Unis, pour l'environnement 9

sur Y de paramètre Θ. La plupart du temps, elle est affine, $f(Y|\Theta) = a \cdot Y + b$ avec $\Theta = (a, b)$. L'élément ϵ est le bruit qui empêche la régression d'être parfaite.

Plusieurs méthodes permettent d'identifier les paramètres Θ optimaux, estimateur du maximum de vraisemblance, estimateur des moments, estimateur robuste et la méthode des moindres carrés. Cette dernière est la plus usuelle. Elle minimise la somme des écarts entre les données X et les données $f(Y|\Theta)$ élevés au carré. Cette somme est notée $S(\Theta)$. Les paramètres optimaux, notés Θ_{optim} sont alors définis comme suit :

$$S(\Theta) = \sum_{i=1}^{n} (x_i - f(y_i|\Theta))^2 \tag{2.22}$$

$$\Theta_{optim} = \underset{\Theta}{argmax} - S(\Theta) \tag{2.23}$$

La fonction $argmax$ permet d'obtenir les valeurs des paramètres maximisant $-S(\Theta)$, c'est-à-dire donnant le plus faible résidu entre les valeurs observées et les valeurs obtenues par régression linéaire.

La qualité de la régression peut être mesurée avec le coefficient de corrélation, r. Il est égal au rapport de la covariance entre X et Y et du produit non nul de leurs écarts types. Il est toujours inclus dans l'intervalle $[-1, 1]$. La valeur 1 indique que la corrélation est parfaite positive. La valeur -1 indique que la corrélation est parfaite négative. Quant à la valeur 0, elle indique l'absence de corrélation. En pratique, une régression est acceptable si la valeur absolue du coefficient de corrélation est supérieure à $0,8$.

$$r = \frac{cov(X, Y)}{\sigma_X \cdot \sigma_Y} \tag{2.24}$$

2.6.3 Modèle de régression

Nous noterons $N_{k,s}(u)$ l'évolution dans le temps du nombre de sessions sur l'environnement k, par pas de s jours et $N_{k,s,p}(u)$ l'évolution du nombre de sessions originaires du pays p sur l'environnement k par pas de s jours. De la même manière, nous noterons $N_s(u)$ l'évolution du nombre de sessions sur tous les environnements par pas de s jours et $N_{s,p}(u)$ l'évolution du nombre de sessions originaires du pays p sur tous les environnements par pas de s jours.

$$N_s(u) = |\{LS_i / s \cdot u \leq datedebut(LS_i) < s \cdot (u+1)\}| \tag{2.25}$$

$$N_{s,p}(u) = |\{LS_i / s \cdot u \leq datedebut(LS_i) < s \cdot (u+1) \wedge geo(LS_i) = p\}| \tag{2.26}$$

$$N_{k,s}(u) = |\{LS_i / s \cdot u \leq datedebut(LS_i) < s \cdot (u+1) \wedge env(LS_i) = k\}| \tag{2.27}$$

$$N_{k,s,p}(u) = |\{LS_i / s \cdot u \leq datedebut(LS_i) < s \cdot (u+1) \wedge env(LS_i) = k \wedge geo(LS_i) = p\}| \tag{2.28}$$

Le nombre de sessions par unité de temps égale la somme des nombres de sessions par unité de temps en provenance des différents pays. Les variables $N_{k,s}(u)$ et $N_{k,s,p}(u)$ sont liées. Dans un premier temps, nous avons tracé les données $N_{k,s}(u)$ et $N_{k,s,p}(u)$, pour chaque environnement et pour différents pays. A vue d'oeil, nous avons observé une surprenante similarité entre ces deux échantillons pour certains pays. La suite présente les résultats de la régression sur les données.

2.6.4 Corrélation entre les environnements

Le tableau 2.10 présente les corrélations entre environnements. La corrélation étant symétrique, la matrice présentée dans ce tableau est symétrique. A première vue, la majorité des coefficients sont faibles. Il existe peu de corrélations entre les environnements. Par contre, les environnements 14, 42 et global sont liés par de fortes corrélations. Le nombre de sessions sur les deux premiers sont du même ordre de grandeur. Toutefois, il est intéressant de constater qu'ils ne font pas partie des environnements les plus actifs alors que leur corrélation avec l'environnement global est élevée. Nous pouvons également noter que les activités enregistrées sur ces deux environnements sont fortement corrélées (facteur de corrélation $0,97$).

2.6.5 Corrélation en fonction de l'origine géographique

Le trafic à destination d'un environnement est originaire de plusieurs pays différents. Toutefois, sur la base d'observations empiriques, nous avons constaté que le trafic originaire de certains pays présente des tendances similaires comparé à l'ensemble du trafic toutes sources confondues. Nous avons voulu vérifier cette hypothèse de façon plus rigoureuse en utilisant le modèle de régression (cf. section 2.6.2).

Nous avons appliqué le modèle de régression linéaire en considérant un, deux ou plus de pays. Les résultats révèlent qu'un modèle de bonne qualité peut être obtenu en considérant uniquement un seul pays. Nous avons identifié trois modèles fournissant les meilleures régressions du nombre total de sessions sur tous les environnements. Ils ont été obtenus en considérant les sessions originaires de la Chine, du Canada et de

environnement j \ environnement i	global	9	13	14	28	31	32	42	62
global	1,00	0,56	0,10	0,86	0,13	0,45	0,51	0,86	0,35
9	0,56	1,00	0,10	0,24	0,16	0,01	0,31	0,23	0,25
13	0,10	0,10	1,00	−0,04	0,12	−0,07	0,20	0,00	0,08
14	0,86	0,24	−0,04	1,00	0,01	0,29	0,18	0,97	0,23
28	0,13	0,16	0,12	0,01	1,00	0,13	0,09	−0,02	−0,08
31	0,45	0,01	−0,07	0,29	0,13	1,00	0,17	0,25	−0,12
32	0,51	0,31	0,20	0,18	0,09	0,17	1,00	0,23	0,25
42	0,86	0,23	0,00	0,97	−0,02	0,25	0,23	1,00	0,29
62	0,35	0,25	0,08	0,23	−0,08	−0,12	0,25	0,29	1,00

TAB. 2.10 – Coefficients de corrélation entre environnements

la France. Les coefficients de corrélation correspondants sont élevés, respectivement $0,87$, $0,86$ et $0,85$. Ils dénotent une bonne régression. Par exemple, le modèle estimé obtenu en considérant les sessions originaires du Canada est défini de la manière suivante :

$$N_s(u) = 9,4708 \cdot N_{s,Canada}(u) + 471,8792 \tag{2.29}$$

La figure 2.17 présente l'évolution du nombre de sessions observé ainsi que le modèle précédent, pour un pas de 1 jour. Notons au passage que le coefficient de corrélation dépend du pas considéré : pour des pas plus grands (une semaine, un mois, ...), les évolutions sont moins bruitées et le coefficient est plus élevé. Nous pouvons constater que le modèle de régression suit globalement bien les données observées. Ces résultats sont d'autant plus surprenants que les sessions originaires de la France et du Canada ne représentent qu'une faible proportion du nombre total de sessions (3% chacun).

Nous avons effectué des analyses similaires en considérant chacun des environnements. Le but est de savoir si des conclusions similaires peuvent être obtenues en confrontant, pour chacun, le nombre total de sessions et le nombre de sessions originaires des différents pays. Les résultats sont présentés dans le tableau 2.11. La première colonne identifie l'environnement. Les six colonnes suivantes contiennent les coefficients de corrélation pour les modèles considérant la Chine, le Canada, la France, les Etats-Unis, la Pologne, l'Espagne comme pays de référence. La dernière colonne contient le coefficient de corrélation du meilleur modèle que nous avons identifié. Le pays correspondant est indiqué entre parenthèses.

Nous pouvons noter que la qualité des régressions en considérant les Etats-Unis comme variable explicative est convenable pour 5 des 8 environnements ($r > 0,8$). Les environnements pour lesquels ce pays ne permet pas d'obtenir un modèle convenable possèdent pourtant une activité importante (cf. tableau 2.9). Le coefficient de corrélation est généralement faible ($r < 0,8$). Cela indique que l'observation effectuée sur la globalité des environnements n'est pas visible d'un point de vue local, environnement par environnement. Cependant, pour la majorité des environnements, le meilleur modèle de régression implique quatre des pays suivants : les Etats-Unis, la Chine, la Pologne et l'Espagne. Ces quatre pays permettent aussi d'aboutir à un bon

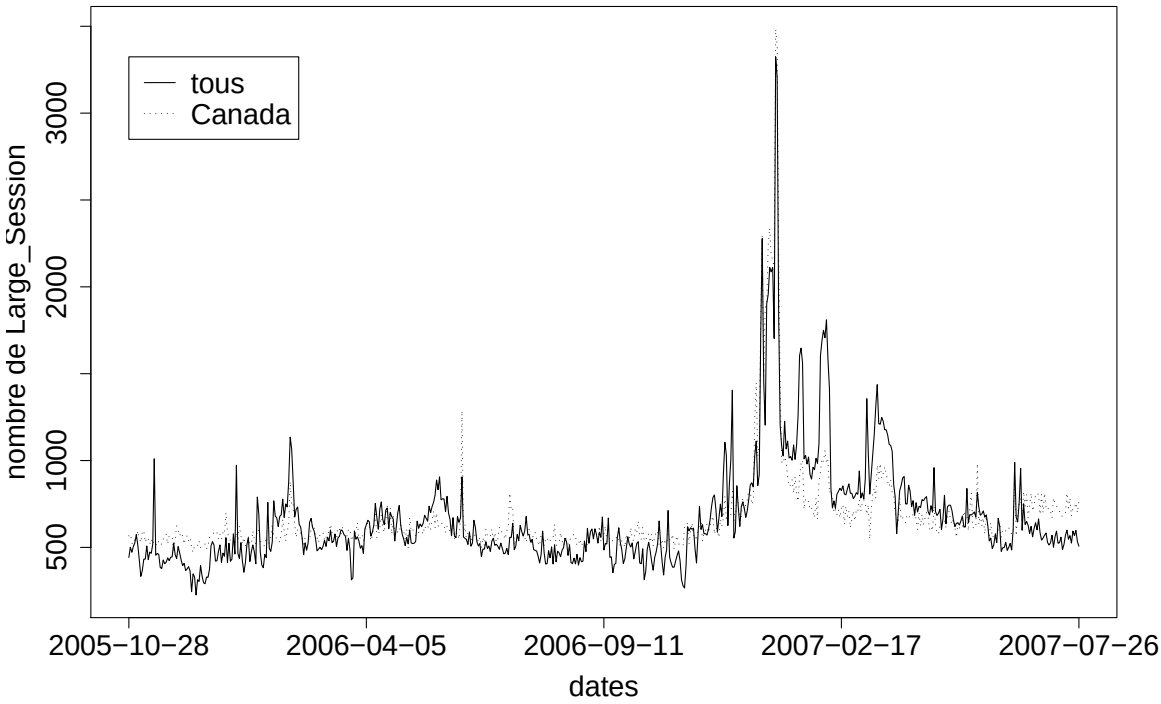


FIG. 2.17 – Evolution du nombre de sessions observé et du nombre de sessions estimé par le modèle de régression, pour tous les environnements

env. k	r Chine	r Canada	r France	r Etats-Unis	r Pologne	r Espagne	meilleur r (pays)
global	0,87	0,86	0,85	0,81	0,61	0,62	0,87 (Chine)
9	0,69	0,65	0,61	0,84	0,51	0,55	0,84 (Etats-Unis)
13	0,56	0,45	0,39	0,63	0,39	0,38	0,63 (Etats-Unis)
14	0,97	0,91	0,93	0,88	0,23	0,41	0,97 (Chine)
28	0,66	0,32	0,25	0,64	0,67	0,28	0,67 (Pologne)
31	0,52	0,57	0,62	0,69	0,66	0,71	0,71 (Espagne)
32	0,58	0,69	0,66	0,83	0,71	0,52	0,83 (Etats-Unis)
42	0,98	0,87	0,93	0,85	0,23	0,46	0,98 (Chine)
62	0,64	0,37	0,41	0,83	0,30	0,36	0,83 (Etats-Unis)

TAB. 2.11 – Résultat de la régression linéaire sur 8 environnements

modèle de régression en considérant l'ensemble des environnements. Par contre, trois environnements s'écartent du lot, 13, 28 et 31, pour lesquels le modèle de régression ne donne pas des résultats intéressants pour aucun des pays considérés.

Les Etats-Unis forment un pays qui peut être utilisé comme variable explicative. Tout d'abord, comme indiqué sur le tableau 2.9, une bonne partie des sessions provient

de ce pays, et ce pour tous les environnements. Ensuite, pour tous les environnements et globalement, les modèles obtenus sont convenables, hormis pour les environnements 13, 28 et 31, pour lesquels toute corrélation avec les pays reste faible de façon générale. Les mêmes remarques peuvent être établies pour la Chine et le Canada. Quant à ce dernier pays, ce qui est remarquable est la faible proportion des sessions originaires de ce pays quelque soit l'environnement considéré. Un pays qui contribue significativement aux attaques en terme de nombre de sessions peut être utilisé pour établir un modèle linéaire acceptable. Il est aussi intéressant de noter que les environnements 14 et 42 présentent des tendances équivalentes par rapport aux corrélations des activités observées sur chacun des environnements toute origine confondue et par rapport aux activités issues de certains pays. Ces résultats renforcent ceux observés au tableau 2.10 montrant une forte corrélation entre les activités de ces deux environnements. Ils sont aussi surprenants dans la mesure où ces deux environnements sont situés dans des sites géographiques différents et ont des adresses distantes.

2.7 Propagation des attaques

Les analyses précédentes ont été menées environnement par environnement et permettent de comprendre les processus d'occurrence des attaques dans le but de mieux paramétrer ces outils. Au delà de ce genre d'analyse, il est utile d'identifier des phénomènes qui reflètent la propagation des attaques à travers les différents environnements. Dans cette section, nous nous intéressons plus précisément à la propagation des attaques sur Internet.

2.7.1 Principe de la propagation des attaques

Nous faisons l'hypothèse qu'une propagation entre deux environnements survient lorsqu'une adresse est observée sur un premier environnement et, par la suite, sur un second environnement. Une telle activité peut résulter de plusieurs activités malveillantes. Nous allons expliquer le principe de la propagation des attaques à travers l'exemple des vers présents sur Internet. La figure 2.18 présente un exemple de ver qui tente de se répandre. Depuis une machine infectée, un ver cible « séquentiellement »des machines choisies aléatoirement. Chacune des machines choisies observe une partie de la tentative de propagation du ver. Sur l'exemple, parmi les 7 machines choisies par le ver, 3 sont des environnements. Sur chacun de ces environnements, nous observons des paquets réseaux qui sont agrégés en sessions. En confrontant les sessions, il nous est possible de reconstituer une partie du parcours suivi par ce ver. Sur l'exemple, nous pouvons affirmer que le ver a ciblé, dans l'ordre, les environnements 1, 2 et 6.

2.7.2 Modèle de propagation

Pour la période identifiée précédemment et ces 8 environnements, nous identifions la liste des sessions. Rappelons qu'une session est caractérisée par l'adresse à son origine, $adr(LS_i)$, l'environnement ciblé, $env(LS_i)$ et la date d'occurrence, $date(LS_i)$. Nous considérons qu'une adresse visite un environnement lorsque nous observons une session réalisée par cette adresse sur cet environnement. Lorsqu'une adresse adr visite

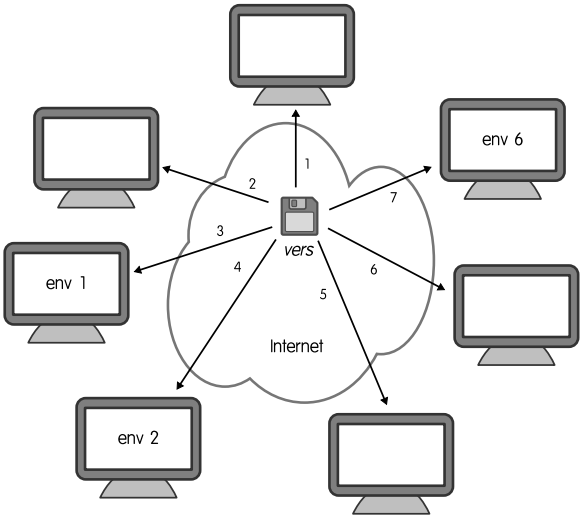


FIG. 2.18 – Exemple de propagation d'un ver

l'environnement e_i et immédiatement après l'environnement e_j, alors nous supposons qu'il y a eu une propagation depuis l'environnement e_i vers l'environnement e_j, issue du processus d'attaque de l'adresse adr.

Une propagation est identifiée par un couple de sessions. L'ensemble des propagations observées, noté PRs, peut être construit de la manière suivante :

$$\begin{aligned} Pr =& \{(ls, ls') \in LS^2 / ls \neq ls' \wedge adr(ls) = adr(ls') \wedge date(ls) \leq date(ls') \wedge \\ & \nexists ls'' \in LS / adr(ls'') = adr(ls) \wedge date(ls) < date(ls'') \leq date(ls')\} \end{aligned} \tag{2.30}$$

Sur la base de l'ensemble des propagations, nous construisons un graphe de propagation. Il s'agit d'un graphe orienté, G, composé de nœuds, l'ensemble V, et de transitions, l'ensemble E, $G = (V, T)$. Un nœud v_j représente un environnement. Une transition $t_{j,k}$ depuis le nœud v_j vers le nœud v_k représente l'ensemble des propagations réalisées depuis l'environnement associé au nœud v_j vers l'environnement associé au nœud v_k. Une transition est donc un couple $t_{j,k} = (v_j, v_k)$. Deux probabilités, notées $P_G(t_{j,k})$ et $Q_G(t_{j,k})$, sont associées à chaque transition.

$P_G(t_{j,k})$ correspond à la probabilité d'occurrence de la propagation associée en tenant compte de l'ensemble des sessions enregistrées pour l'environnement v_j. La probabilité $P_G(t_{j,k})$ est le rapport entre le nombre de propagations réalisées entre les environnements v_j et v_k et le nombre de propagations réalisées depuis l'environnement v_j. $Q_G(t_{j,k})$ correspond à la probabilité d'occurrence de la propagation associée en tenant compte uniquement des sessions correspondant à des propagations en sortie de v_j La probabilité $Q_G(t_{j,k})$ est le rapport entre le nombre de propagations réalisées entre les environnements v_j et v_k et le nombre de propagations réalisées en sortie de l'environnement v_j.

$$P_G(t_{j,k}) = \frac{|\{(ls, ls') \in PRs/env(ls) = env(v_j) \wedge env(ls') = env(v_k)\}|}{|\{(ls, ls') \in PRs/env(ls) = env(v_j)\}|} \tag{2.31}$$

$$Q_G(t_{j,k}) = \frac{|\{(ls, ls') \in PRs/env(ls) = env(v_j) \wedge env(ls') = env(v_k)\}|}{|\{(ls, ls') \in PRs/env(ls) = env(v_j) \wedge ls \neq ls'\}|}, \quad \forall j \neq k \tag{2.32}$$

Au stade actuel de nos travaux, le temps écoulé entre les visites successives vers les différents environnements n'est pas pris en compte. Introduire des mesures liées au temps qui s'est écoulé entre deux transitions successives pourrait nous permettre d'affiner ce modèle. En particulier, nous pourrions considérer qu'une propagation se produit uniquement si le temps séparant deux visites à deux environnements différents est inférieur à un seuil (par exemple de l'ordre de quelques heures voire quelques minutes).

2.7.3 Illustration

Les résultats obtenus avec les 8 environnements sélectionnés dans notre étude sont présentés dans cette section. Le tableau 2.12 contient le nombre de propagations réalisées entre chaque paire d'environnements. Sur la base de ce tableau, nous pouvons calculer les différentes valeurs des probabilités $P_G(t_{j,k})$ et $Q_G(t_{j,k})$. Les résultats sont présentés aux figures 2.13 et 2.14. A titre d'illustration, la figure 2.19 fournit une représentation graphique des probabilités $Q_G(t_{j,k})$.

environnement j \ environnement i	9	13	14	28	31	32	42	62
9	8029	744	2098	828	2258	2562	1411	1733
13	700	768	802	178	478	1747	430	249
14	2161	700	3792	835	1042	1424	4486	1144
28	679	272	768	1872	284	488	663	316
31	2342	476	1135	362	8415	1443	746	728
32	3026	1149	1376	443	1209	6032	934	2783
42	1186	612	4875	457	765	1071	2948	760
62	2658	204	1138	343	726	1532	840	2662

TAB. 2.12 – Nombre de propagations entre environnements

Tout d'abord, nous constatons que le graphe est fortement connexe et complet. Quelque soit le couple d'environnements considéré, il existe deux arcs orientés entre ces deux environnements. Cependant, les probabilités ne sont pas homogènes : aucun nœud ne possède des arcs aux probabilités du même ordre de grandeur.

En considérant les valeurs reportées dans le tableau 2.13, nous pouvons constater que généralement, les probabilités de rester dans l'environnement sont faibles. La valeur la plus élevée est de l'ordre de 54%, pour l'environnement 31. Ceci laisse supposer que les attaquants ont tendance à visiter plusieurs environnements successivement (c'est le cas, par exemple, lors des activités de *scan*).

environnement j \ environnement i	9	13	14	28	31	32	42	62
9	41%	4%	11%	4%	11%	13%	7%	9%
13	13%	14%	15%	3%	9%	33%	8%	5%
14	14%	4%	24%	5%	7%	9%	29%	7%
28	13%	5%	14%	35%	5%	9%	12%	6%
31	15%	3%	7%	2%	54%	9%	5%	5%
32	18%	7%	8%	3%	7%	36%	6%	16%
42	9%	5%	38%	4%	6%	8%	23%	6%
62	26%	2%	11%	3%	7%	15%	8%	26%

TAB. 2.13 – Probabilités $P_G(t_{j,k})$ pour les 8 environnements

environnement j \ environnement i	9	13	14	28	31	32	42	62
9	–	6%	18%	7%	19%	22%	12%	15%
13	15%	–	17%	4%	10%	38%	9%	5%
14	18%	6%	–	7%	9%	12%	38%	10%
28	20%	8%	22%	–	8%	14%	19%	9%
31	32%	7%	16%	5%	–	20%	10%	10%
32	28%	11%	13%	4%	11%	–	9%	25%
42	12%	6%	50%	5%	8%	11%	–	8%
62	36%	3%	15%	5%	10%	21%	11%	–

TAB. 2.14 – Probabilités $Q_G(t_{j,k})$ pour les 8 environnements

En regardant plus particulièrement les probabilités du tableau 2.14, pour apprécier en particulier le poids relatif des propagations entre les environnements, nous pouvons observer une forte dépendance entre certains environnements. C'est le cas des environnements 42 et 14. Par exemple, la probabilité de visiter 14 à partir de 42 est de l'ordre de 50% à comparer à 5% pour la probabilité de visiter 28 à partir de 42. Cependant, les propagations entre deux environnements ne sont pas nécessairement du même ordre de grandeur dans les deux sens (la probabilité de propagation de 14 vers 42 est de l'ordre de 38%).

Nous pouvons constater aussi que les probabilités de passer d'un environnement à certains environnements restent faibles. C'est le cas par exemple des environnements 13 et 28. Ceci peut résulter du fait que ces deux environnements possèdent l'activité la plus faible.

Enfin, il est aussi intéressant d'observer que certains environnements (31 et 62) présentent des comportements équivalents en sortie (c'est-à-dire que les probabilités $Q_G(t_{31,k})$ et $Q_G(t_{62,k})$ sont du même ordre de grandeur).

2.8 Conclusion

Dans ce chapitre, nous avons présenté une méthodologie permettant l'exploitation des données issues des pots de miel basse interaction déployés dans le cadre de la

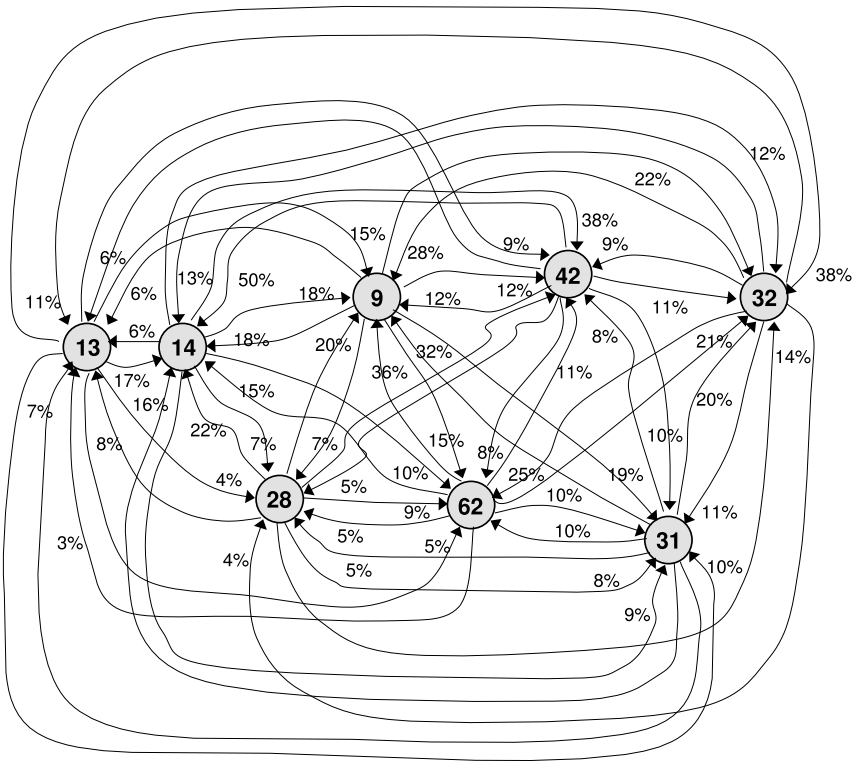


Fig. 2.19 – Graphe de propagation associé aux 8 environnements

plate-forme Leurre.com pour élaborer des modèles stochastiques caractérisant le processus d'occurrence des attaques observées et leur propagation. Nous avons étudié plus particulièrement les distributions des intervalles de temps entre attaques et l'évolution du nombre d'attaques par unité de temps sur différents environnements toutes sources confondues et en fonction de leur origine géographique. Nous avons aussi étudié des corrélations entre les activités observées sur plusieurs environnements, ainsi que les propagations d'attaques entre ces environnements.

La méthodologie proposée s'appuie sur des outils statistiques permettant en particulier d'identifier des périodes de silence suspectes susceptibles de biaiser les analyses si elle ne sont pas identifiées en tant que telles, et de guider la sélection d'un ensemble de données et d'environnements pertinent pour élaborer des modèles et effectuer des analyses comparatives significatives.

La modélisation des intervalles de temps entre attaques a permis d'identifier une distribution de probabilité constituée d'un mélange d'une loi de Pareto généralisée et d'une loi de Weibull, qui est bien adaptée pour décrire les activités d'attaques observées sur les différents environnements étudiés. Cette distribution met en évidence deux types de trafics malveillants qui cohabitent : des attaques en rafales et des attaques monotones plus espacées dans le temps. Ces modèles peuvent être utilisés pour générer du trafic malveillant représentatif des activités observées sur les pots de miel basse interaction. Cependant, l'utilisation de ces modèles dans une optique prévisionnelle ou pour améliorer l'efficacité des mécanismes de détection reste un problème ouvert.

Par ailleurs, les analyses de corrélation et les modèles de régression linéaire étudiés dans ce chapitre ont permis d'identifier certains comportements surprenants, concernant en particulier la corrélation entre l'évolution du trafic toutes sources confondues et le trafic provenant de certains pays uniquement, qui ne représentent pas une part importante du trafic observé. Ces analyses ont aussi permis d'identifier des environnements de pots de miel qui présentent des évolutions et des comportements similaires alors qu'ils ne sont pas localisés géographiquement au même endroit et ne possèdent pas des adresses proches. Nous avons aussi montré que les analyses de propagation sont aussi utiles pour caractériser les processus d'attaque observés.

Ces observations montrent que les modèles que nous avons étudiés apportent des éclairages complémentaires aux analyses présentées dans d'autres travaux effectués dans ce domaine.

Les données que nous avons considérées dans ce chapitre sont issues de pots de miel basse interaction. Elles permettent d'analyser les comportements des attaquants qui interrogent les pots de miel sans pour autant pouvoir pénétrer au cœur du système. Nous sommes aussi intéressés par l'analyse du comportement des attaquants qui ont réussi à prendre le contrôle d'un système cible. Les chapitres suivants ont pour objectif de répondre à ce besoin en utilisant des pots de miel haute interaction.

Chapitre 3

Développement d'un pot de miel haute interaction

3.1 Introduction

Les données et analyses précédentes sont intéressantes pour étudier l'intensité et l'évolution des attaques sur Internet, pour l'élaboration de modèles. Cependant, les pots de miel utilisés ne laissent pas la possibilité aux attaquants de s'introduire au cœur du système. Le comportement des attaquants au sein du système est important à analyser pour affiner nos connaissances sur leur manière d'opérer. Un pot de miel d'interaction plus élevée est nécessaire. Il doit, en particulier, nous permettre d'observer l'évolution de l'attaquant dans le système. Le but de ce chapitre est de présenter une architecture de pot de miel haute interaction répondant à ce besoin. Dans un premier temps, nous caractérisons les pots de miel haute interaction et nous en étudions différentes implémentations. Ensuite, nous présentons notre implémentation de pot de miel.

3.2 Caractéristiques des pots de miel haute interaction

A l'origine, les pots de miel se définissaient essentiellement par leur niveau d'interaction, haute ou basse. Depuis, le concept de pot de miel à beaucoup évolué. Récemment, des implémentations intermédiaires ont fait leur apparition, rendant la notion d'interaction insuffisante pour caractériser un pot de miel (cf. section 1.4.2.4 du chapitre 1). Afin de confronter les différentes implémentations de pots de miel haute interaction, nous avons identifié quatre points essentiels qui les caractérisent : la transparence, l'observabilité, la flexibilité et la nature. Quant au niveau d'interaction, il peut se définir comme la composition de ces quatre caractéristiques.

3.2.1 La transparence

Plus il est nécessaire d'avoir recours à des techniques avancées pour démasquer un pot de miel, plus ce pot de miel possède un haut niveau de transparence. Ces techniques peuvent être des plus simples, comme la lecture de fichier de journalisation, ou des plus compliquées, comme l'étude de la latence introduite par d'éventuels méca-

nismes d'observation. Pour obtenir un bon niveau de transparence, les modifications à apporter au système d'exploitation doivent être minimes et localisées à des zones difficiles d'accès (dans la mémoire du noyau, par exemple).

3.2.2 L'observabilité

L'observabilité est la capacité d'un pot de miel à récupérer des informations sur l'activité des intrus s'y étant introduits. L'observabilité est le pendant de la transparence. Pour qu'un pot de miel possède un haut niveau d'observabilité, il faut mettre en place des mécanismes de collecte souvent très diversifiés voire très lourds. Cette surcharge, entraînée par ces mécanismes, diminue considérablement le niveau de transparence. Par conséquent, un compromis entre l'observabilité et la transparence doit être fixé.

3.2.3 La flexibilité

La flexibilité d'un pot de miel correspond à sa capacité à s'adapter aux besoins d'une étude. Un pot de miel ayant un haut niveau de flexibilité permet de reconfigurer rapidement l'instrumentation. Plus ce niveau est élevé, plus le pot de miel sera efficace dans une démarche itérative : c'est-à-dire que nous pouvons, à chaque étape de l'apprentissage sur le comportement des attaquants, le reconfigurer facilement pour offrir plus de liberté à l'intrus, afin d'enrichir les observations.

3.2.4 La nature

Un pot de miel peut être soit physique soit virtuel. Un pot de miel est qualifié de physique lorsqu'il peut être installé sur sa propre machine physique. Il faut donc autant de machines physiques que de pots de miel physiques. Un pot de miel virtuel doit, quant à lui, être installé comme invité au dessus d'un système d'exploitation hôte. Cette dernière solution présente l'avantage de n'utiliser qu'une seule machine physique pour la mise en service de plusieurs pots de miel virtuels. Les pots de miel physiques sont très coûteux. Pour chaque système d'exploitation que l'on souhaite utiliser en tant que pot de miel, une machine doit être installée et administrée. A ces problèmes, vient se greffer celui de la collecte d'informations issues des activités des intrus. Lors du rapatriement de ces informations, les connexions réalisées doivent être cachées des intrus pour ne pas perturber leur comportement. Pour contourner ces problèmes, les différentes implémentations de pot de miel se sont rapidement orientées vers des techniques de virtualisation. Elles permettent d'exécuter sur une seule machine physique plusieurs systèmes d'exploitation, voire plusieurs applications. Entre autres, elles sont utilisées afin de minimiser les dépenses en matériel, de faciliter les déploiements, de tester des implémentations et d'isoler, pour des raisons de sécurité, l'environnement d'éxécution d'une application. Les techniques de virtualisation les plus connues sont[URLb] :

- *isolateur* : un isolateur est un logiciel permettant de cloisonner l'exécution de processus. Au même instant, il permet d'exécuter plusieurs fois une application prévue pour être exécutée une seule fois par machine. Exemples d'implémentation : `Linux-Vserver`[Gél] et `OpenVZ`[Sav].

- *paravirtualisation* : un système d'exploitation joue le rôle d'hyperviseur dans le but d'exécuter des systèmes d'exploitation invités. Ces derniers ont conscience d'être virtualisés et peuvent aussi utiliser des services du système d'exploitation hôte. Exemples d'implémentation : XEN[Uni], UML[Dik06] et KVM[URLa].
- *virtualisation complète* : un logiciel permet de virtualiser le matériel pour exécuter des systèmes d'exploitation invités. Ces derniers n'ont pas conscience d'être virtualisés. Exemples d'implémentation : QEMU[Bel], VMWARE[VMw] et KVM.

Les micro-processeurs récents proposent des instructions d'aide à la virtualisation. Une partie non négligeable des tâches de virtualisation qui incombait au niveau logiciel est reportée au niveau matériel, améliorant sensiblement les performances. Les implémentations basées sur les techniques de paravirtualisation et de virtualisation complète peuvent profiter de ces instructions. KVM et VMWARE ont été développés dans cette optique.

3.3 Implémentations de pots de miel haute interaction

Dans cette section, nous présentons différentes implémentations de pot de miel haute interaction. Chacune est dépeinte en fonction des caractéristiques précédentes.

3.3.1 Un pot de miel avec VMWARE

Dans [Hon03], les auteurs proposent d'utiliser le logiciel VMWARE pour la mise en place d'un pot de miel virtuel. Ce document constitue un guide pour le déploiement d'une architecture de pot de miel avec VMWARE. Tout d'abord, le logiciel VMWARE est installé et configuré. Ensuite, une partie dédiée au contrôle des données est abordée. Pour finir, les auteurs présentent les mécanismes de capture des données. Le pot de miel est déployé sur une machine hôte de type GNU/LINUX. Une machine invitée est installée sur cette machine hôte, par le biais du logiciel VMWARE. Le contrôle des données est assuré par l'utilisation de la commande du pare-feu de Linux, `iptables`. Plus précisément, c'est la fonctionnalité de limitation des échanges (en anglais, *rate-limiting*) qui est utilisée : le nombre de connexions sortantes (initiées depuis le pot de miel, à destination de Internet) est limité. Quant à la capture des données, elle est réalisée par l'outil `snort`[CBFF03], un renifleur de réseau, et la fonctionnalité de journalisation de `iptables`.

Cette architecture dispose d'un bon niveau de transparence : les systèmes d'exploitation invités ne sont pas modifiés. Les seules possibilités pour l'attaquant de deviner la présence du pot de miel sont l'utilisation des méthodes statistiques pour découvrir la politique pratiquée par le pare-feu et l'analyse de la présence des empreintes laissées par VMWARE sur les machines invitées[Kor].

L'observabilité, par contre, n'est pas très élevée. Effectivement, les méthodes de collecte de données ne permettent pas de récupérer des informations dans les connexions chiffrées issues de l'accès au service `ssh`, par exemple. Par contre, ce pot de miel est très flexible car il est possible d'utiliser toute sorte de machine invitée.

3.3.2 UML comme pot de miel

Dans [HM03], les auteurs présentent une architecture de pot de miel haute interaction, de nature virtuelle, basée sur UML (USER-MODE-LINUX) et PRELUDE. UML utilise la technique de paravirtualisation pour permettre d'exécuter, sur un hôte GNU/LINUX, des systèmes d'exploitation GNU/LINUX invités. PRELUDE est un système de détection d'intrusions hybride composé d'un manager et de plusieurs sondes :

- `Manager` : le manager est chargé de centraliser les informations collectées par les différentes sondes. Lors d'une alerte, le manager peut avertir un dispositif de contre mesure. Il peut aussi avertir un manager global, permettant ainsi de déployer une architecture hiérarchique.
- `Prelude-NIDS` : cette sonde collecte des traces réseau issues d'activités malveillantes. Dans [HM03], il est indiqué d'exécuter ce processus sur la machine hôte.
- `Prelude-LML` : cette sonde scrute régulièrement les fichiers de journalisation disponibles sur le système d'exploitation invité. Elle doit être exécutée sur ce dernier, en tâche de fond.

Afin de donner l'illusion à l'attaquant que les machines invitées sont régulièrement utilisées, des scripts sont régulièrement exécutés pour volontairement provoquer de l'activité. Dans l'ensemble, cette architecture possède une bonne observabilité et une bonne flexibilité, par l'emploi de PRELUDE. Toutefois, PRELUDE, bien qu'apportant beaucoup de flexibilité, n'a pas été développé dans l'optique d'être installé sur un pot de miel. Aucun mécanisme de furtivité n'a été mis en place pour cacher la présence du processus `Prelude-LML`. Ce dernier est facilement détectable par les attaquants. La transparence s'en voit beaucoup affectée, ce qui constitue le principal inconvénient de cette architecture. La figure 3.1 présente cette architecture.

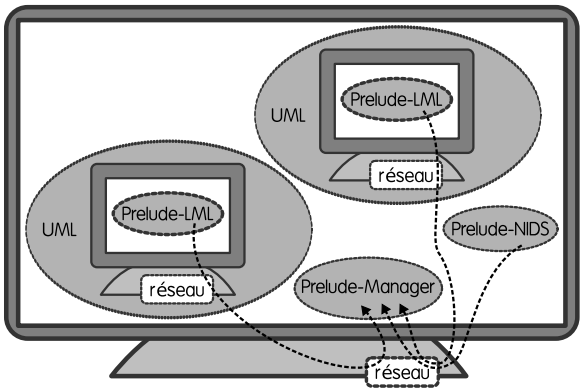


FIG. 3.1 – Architecture d'un pot de miel basé sur PRELUDE et UML

3.3.3 SEBEK

SEBEK[Pro03] est un pot de miel physique. Il est composé d'un serveur et d'autant de clients que de pots de miel. Le serveur contient une base de données qui accueille les données collectées sur les pots de miel. Sur chacun des clients – donc des pots de miel – est installé dans le noyau un module qui permet d'intercepter les appels système.

Un appel système est une primitive du noyau permettant de rendre des services aux programmes exécutés au niveau de l'espace utilisateur. Le but est d'autoriser un programme exécuté en espace utilisateur de solliciter un service de l'espace noyau. Pour ce faire, le programme renseigne un registre du microprocesseur avec le numéro du service et lance une interruption logicielle (`int/sysenter`). Cette interruption, traitée au niveau de l'espace noyau, va permettre l'exécution d'un code du noyau. Ce code va chercher dans une table, la table des appels système, l'adresse de la routine correspondant au numéro renseigné dans le registre du microprocesseur, par le programme. Le schéma 3.2 présente le mécanisme d'interception d'un appel système.

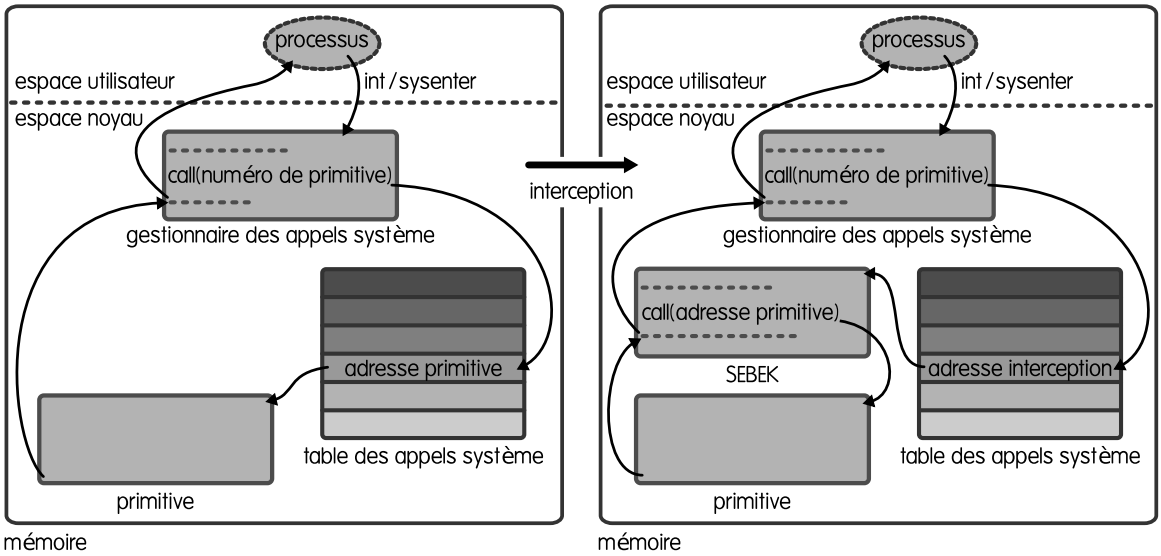


Fig. 3.2 – Mécanisme d'interception d'un appel système

Sebek intercepte de cette manière plusieurs appels système (`read`, `write`, ...). L'interception de ces appels système permet de récupérer des informations intéressantes sur l'activité du pot de miel. Par exemple, l'appel système `read` est utilisé pour lire toute forme d'information, des octets circulant sur le réseau aux octets stockés dans des fichiers. En l'interceptant, il est alors possible de reconstituer ce que l'intrus voit sur son terminal et d'identifier les fichiers qui ont sollicité l'intérêt de l'intrus. Les informations collectées par ces interceptions sont envoyées depuis le pot de miel vers une base de données, à travers le réseau. Les connexions permettant ce rapatriement sont cachées de l'intrus par une modification de la partie gestion du réseau du noyau. Ce pot de miel possède l'avantage d'être facilement portable. Il peut être installé aussi bien sur des systèmes d'exploitation Gnu/Linux que Windows. Son niveau d'observabilité est, quant à lui, très élevé. Ceci est principalement dû aux interceptions des appels système. Concernant sa transparence, il utilise des techniques évoluées mais connues des attaquants. Aussi, les contre-mesures à ces techniques sont elles aussi bien connues[Cor04, DHK04]. Sebek est donc facilement détectable.

3.3.4 Uberlogger

Uberlogger[AGJ05] est un pot de miel haute interaction de nature physique. Il

opère, dans les grandes lignes, de la même manière que SEBEK. Un module du noyau est chargé en mémoire pour intercepter des appels système. Les informations collectées par ces interceptions sont envoyées à travers le réseau et à destination d'une base de données. Dans cette implémentation, l'accent a été placé sur les mécanismes permettant un haut niveau de transparence. Deux de ces mécanismes les plus intéressants sont (1) la modulation de la quantité des données envoyées sur le réseau en fonction de la charge du pot de miel, et (2) la gestion des *capabilities*, jeton utilisé par un processus pour prouver qu'il est autorisé à exécuter certaines tâches, permettant de restreindre les possibilités d'action de l'intrus. Cette approche rend plus difficile les attaques statistiques pour démasquer la présence d'un outil d'observation.

3.4 Architecture du pot de miel haute interaction

L'objectif que nous nous sommes fixés vise à étudier le comportement des attaquants humains sur Internet. Nous avons besoin d'un mécanisme d'observation. Dans cette section, nous présentons nos objectifs. Pour chacun d'eux, nous exposons une architecture permettant de l'atteindre. L'architecture générale est ensuite présentée.

3.4.1 Les objectifs et les données à collecter

Nous souhaitons disposer d'un moyen d'observer le comportement des attaquants humains pendant leur intrusion dans un système informatique. Pour ce faire, nous devons répondre à trois questions : *Comment attirer les attaquants humains ? Comment collecter des informations sur leurs activités ?* et *Comment contrôler ces activités ?*

Nous souhaitons avoir la capacité d'observer le comportement des attaquants humains au cœur d'un système informatique. Pour ce faire, l'emploi d'un pot de miel haute interaction est tout indiqué. Les attaquants, depuis Internet, peuvent profiter des vulnérabilités proposées sur le pot de miel haute interaction pour y entrer. Toutefois, un tel dispositif peut capturer différents types d'attaquants : aussi bien des êtres humains que des outils automatiques. Plusieurs études ont été réalisées pour étudier, par exemple, les vers qui se propagent sur Internet. Entre autres, dans [DJG07], les auteurs utilisent des pots de miel pour identifier des motifs d'attaque des vers sur Internet, dans le but d'améliorer les moyens de détection de ces activités malveillantes. L'approche par pot de miel a aussi permis d'analyser le comportement des attaquants qui désirent créer des *botnets*, comme expliqué dans [RJMT06] et [ZC06]. Pour notre part, nous avons choisi de cibler notre étude sur des attaquants humains. Afin de limiter le nombre d'attaquants automatiques observés, nous allons les filtrer à l'entrée du pot de miel au niveau des vulnérabilités proposées. Plus précisément, nous allons rendre l'accès au pot de miel plus facile aux attaquants humains en utilisant un système d'exploitation GNU/LINUX et en proposant une vulnérabilité des plus anciennes : des comptes utilisateur avec des mots de passe simples à deviner, accessibles depuis Internet par le service `ssh`. Etant donné qu'il est difficile d'envisager un outil automatique capable de mener une activité cohérente via le service `ssh` dans un interpréteur de commandes, nous minimisons le nombre d'observations d'attaquants automatiques. Pour l'observation du comportement des attaquants humains, il est essentiel que les pots de miel offrent un plus haut niveau d'interaction. Nous avons opté pour la solution visant à mettre à disposition des attaquants plusieurs machines,

et non une seule. Administrer plusieurs machines physiques est un travail fastidieux. L'utilisation de techniques de virtualisation permet de n'avoir à administrer qu'une seule machine physique, sur laquelle sont exécutés plusieurs systèmes d'exploitation. Notre choix s'est porté sur l'emploi d'une de ces techniques (cf. section 3.2.4). Nous utilisons donc une implémentation de nature virtuelle, plus simple à mettre en place pour l'administration de plusieurs machines.

Nous nous plaçons dans un cycle d'apprentissage sur le comportement des attaquants humains. Cette implémentation doit donc proposer un bon niveau de flexibilité. Elle doit aussi proposer un niveau d'observabilité adapté aux humains, tout en assurant une bonne transparence. Son instrumentation est donc nécessaire pour collecter des informations qui nous semblent pertinentes pour étudier le comportement des attaquants. Ces informations sont atomiques et elles consitutent les unités d'observation. Leur combinaison en information de plus haut niveau doit permettre de reconstituer les scénarios suivis par les attaquants humains. Nous avons donc identifié trois sources d'informations. Tout d'abord, les couples (nom d'utilisateur, mot de passe) tentés par l'attaquant pour se connecter sont importants à collecter. Ensuite, les caractères circulant dans les connexions `ssh` sont une bonne source d'information. Ils permettent de déterminer les commandes exécutées par l'attaquant et de savoir ce que cet attaquant voit sur son ordinateur. A travers une connexion `ssh`, une exécution d'un programme s'effectue en saisissant le nom complet du fichier exécutable correspondant. Or, certains raccourcis clavier permettent aux utilisateurs d'exécuter ces programmes plus rapidement, en utilisant des scripts ou en ne saisissant qu'une partie du nom du fichier. Pour pallier ce problème, nous capturons les noms des fichiers exécutés par le système d'exploitation.

Un point délicat à traiter, concernant les pots de miel haute interaction, est le confinement des activités illicites. Lorsqu'un attaquant a réussi à s'introduire dans le pot de miel, il ne doit pas disposer d'un moyen d'accès vers l'extérieur depuis ce pot de miel, pour des raisons légales. Autrement dit, nous devons contrôler le phénomène de rebonds. Nous pourrions interdire les connexions sortantes, tout simplement. L'inconvénient de cette approche est qu'elle masque des informations importantes. Notamment, il serait impossible de situer nos observations dans le scénario suivi par l'attaquant, qui peut être compliqué. Une alternative consiste à employer des techniques permettant de limiter le trafic sortant (*rate-limiting*). Cette technique permet d'accepter les connexions sortantes tant qu'elles restent en deça d'une certaine limite. Si nous l'appliquons aux pots de miel, chaque attaquant se voit affecter la possibilité de réaliser un certain nombre de connexions sortantes mais pas plus. De cette manière, plus d'informations sont obtenues. Mais cela reste insuffisant et surtout ne résoud pas les problèmes de légalité juridique. De plus, les implémentations existantes, souvent intégrées dans les pare-feux, ne sont pas assez souples pour être utilisées avec des pots de miel. Pour résoudre le problème des rebonds, une solution possible serait de disposer d'une technique permettant de donner à l'attaquant l'illusion qu'il peut effectivement "rebondir" depuis notre pot de miel, grâce à un mécanisme de redirection. Le principe consiste à faire en sorte que les connexions sortantes de notre pot de miel soient possibles en "apparence", mais uniquement si redirigées vers d'autres pots de miel. L'originalité de notre approche réside dans le fait que cette redirection se fait à la volée, dynamiquement, grâce à un module inséré dans le noyau Linux. Lorsqu'un attaquant, depuis notre pot de miel, se met à la recherche de nouvelles cibles sur

Internet (par des scans de réseau par exemple), nous faisons en sorte que certains de ces scans aboutissent en les redirigeant à la volée sur un autre de nos pots de miel. L'attaquant peut alors entreprendre de poursuivre ses activités.

A notre sens, les implémentations précédentes ne suffisent pas pour atteindre nos objectifs. VMware, en tant que pot de miel[Hon03], ne propose pas de moyens pour collecter des informations sur les commandes lancées par les attaquants au sein du pot de miel. L'implémentation basée sur Uml et présentée dans la section 3.3.2[HM03] ne dispose pas d'assez de transparence pour être cachée des attaquants. Quant aux pots de miel Sebek et Uberlogger, ils s'approchent beaucoup de l'implémentation qui nous convient mais ils ne proposent aucune solution au problème des rebonds. Nous avons estimé que l'effort à mettre en œuvre pour créer une nouvelle implémentation est moins important que pour modifier les implémentations existantes. Toutes ces constatations nous ont amenés à considérer le développement de notre propre implémentation de pot de miel.

3.4.2 L'observation des activités des attaquants

Le but d'un pot de miel est la collecte d'informations sur les activités des attaquants. Pour réaliser cette collecte, le pot de miel est instrumenté. En ce qui concerne notre pot de miel haute interaction, l'instrumentation doit être pensée de manière à collecter, entre autres, les caractères qui transitent dans les connexions `ssh`. Pour savoir de quelle manière cette collecte doit être menée, nous nous penchons sur ce que nous savons, a priori, des attaquants humains. Ensuite, nous rappelons le fonctionnement d'une connexion `ssh`. Pour finir, nous présentons différentes manières de collecter les caractères ainsi que celle que nous adoptons.

3.4.2.1 Remarques préliminaires

Les attaquants, lorsqu'ils pénètrent un système, ne cherchent pas forcément à éveiller les soupçons. Cacher leur activité peut même être l'un de leurs principaux soucis. Si le moyen de cacher leur connexion leur est offert, ils l'utiliseront probablement. Les attaquants peuvent donc communiquer avec leurs cibles au travers de connexions chiffrées. De la sorte, ils masquent un minimum leurs activités. Leurs cibles proposent souvent en standard les librairies nécessaires pour réaliser ce genre de connexions. Citons, par exemple, la librairie `SSL` sur les systèmes d'exploitation Unix. Dans le cas contraire, il peut être aisé pour les attaquants de télécharger et d'exécuter sur leurs cibles – et dans les prémices de leurs attaques – un programme qui embarque ces librairies. Les connexions chiffrées rendent difficile l'étude de l'activité des attaquants. Or, les renseignements qui transitent au travers de ces connexions – et qui sont interprétés par le pot de miel – sont d'une grande importance pour collecter des informations sur les activités des attaquants au sein du pot de miel. Afin d'assurer un minimum d'observabilité, nous devons considérer ce problème et trouver une manière de visualiser le contenu de ces connexions.

3.4.2.2 Fonctionnement de `ssh`

Avant d'entrer dans le cœur de cette problématique, rappelons le fonctionnement du service `ssh`. La figure 3.3 donne la séquence suivie pour l'échange de clé utilisée par

ce protocole[YL06]. Tout d'abord, un client, qui désire se connecter à distance sur un serveur, envoie une requête auprès du serveur. Se déroule alors l'étape de négociation de la méthode à utiliser pour le chiffrement des données. Ensuite, le serveur envoie des informations sur le corps dans lequel le chiffrement sera réalisé, p et g. Le client, à la réception de ces données, génère aléatoirement une variable x. En aucun cas cette valeur ne doit circuler en clair sur le réseau. Par contre, la valeur $g^x \mod p$ est envoyée. Que cette dernière valeur circule en clair n'est pas un problème : une bonne partie de la sécurité de `ssh` repose sur la difficulté – voire l'impossibilité – de retrouver la valeur x à partir de $g^x \mod p$, p et g (problème du logarithme discret). Le serveur, dès reception de $g^x \mod p$, génère à son tour une valeur aléatoire, y, et envoie $g^y \mod p$ au client. Dès à présent, la valeur secrète $g^{xy} \mod p$, nommée clé de session, peut être calculée par les deux protagonistes de la communication. La clé de session sera employée en tant que clé de chiffrement dans le protocole symétrique de chiffrement négocié précédemment. Cet échange se nomme l'échange de clé Diffie-Hellman[MW98].

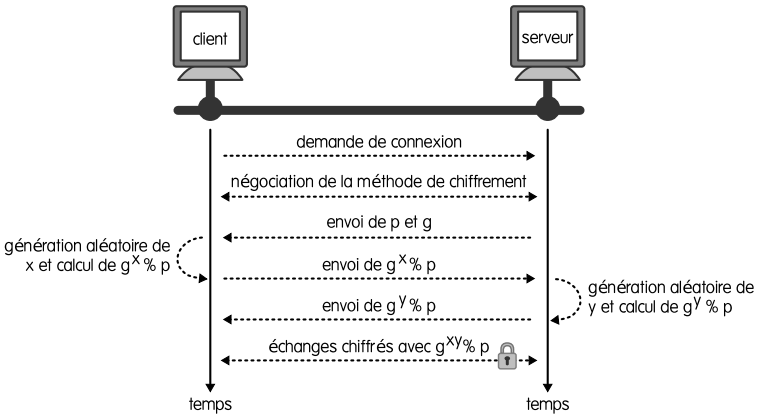


FIG. 3.3 – L'échange de clé Diffie-Hellman

3.4.2.3 Moyens d'observation des connexions `ssh`

Pour visualiser le contenu des connexions chiffrées, nous pouvons adopter plusieurs solutions différentes. La première consiste à réaliser une instrumentation en modifiant directement les logiciels installés sur le pot de miel, tels que le système d'exploitation. Cette approche, souvent utilisée, présente l'inconvénient majeur d'être facilement détectable par les attaquants. Par conséquent, elle peut modifier le comportement des attaquants. La seconde solution consiste à se baser uniquement sur les traces réseau afin d'en déduire les activités au cœur du pot de miel. Elle se base sur des notions de cryptanalyse, et le plus souvent sur les biais statistiques introduits par les algorithmes de chiffrement dans les programmes permettant les connexions chiffrées. La présentation de `SEBEK`, dans la section 3.3, nous a familiarisé avec la première solution. Penchons-nous sur la seconde solution.

Dans [SWT01], l'auteur propose une méthode statistique d'étude des connexions chiffrées, dans le but d'inférer le contenu de ces connexions. Cette méthode est illustrée

avec le service `ssh`, mais peut être adaptée à d'autres types de connexions chiffrées. La méthode présentée dans [SWT01] s'appuie sur les constatations suivantes :

- Un client `ssh` envoie un paquet réseau par touche pressée par l'utilisateur, pour assurer un haut niveau d'interactivité. Autrement dit, l'émission d'un paquet réseau correspondant à la pression d'une touche n'est pas temporisée.
- La durée entre les émissions de deux paquets consécutifs, du côté de l'utilisateur, dépend des deux touches pressées. Par exemple, cette durée est très faible pour la séquence de touches (v, o) (emploi d'un doigt de chaque main) et plus élevée pour la séquence de touches (v, r) (emploi du même doigt de la main gauche).

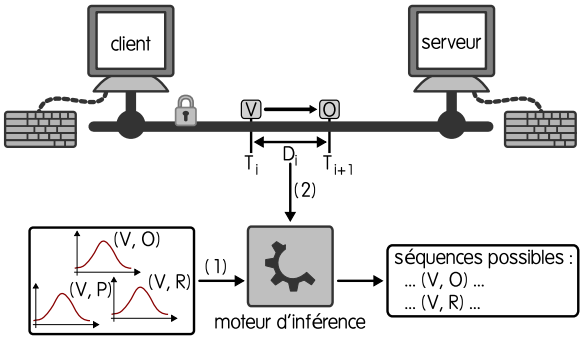


FIG. 3.4 – Méthode d'inférence du contenu de connexions chiffrées

La figure 3.4 illustre cette méthode. En disposant (1) des distributions de probabilités des durées entre émissions de deux touches consécutives pour chaque couple de touches et (2) des durées entre les émissions des paquets consécutifs, il est alors possible de déterminer le contenu de ces paquets, avec plus ou moins de certitude. Un attaquant peut observer un réseau, collecter les paquets échangés lors d'une connexion sécurisée entre un serveur et un utilisateur et utiliser cette méthode pour en extraire des informations intéressantes (des mots de passe par exemple). Nous pourrions utiliser ces méthodes – à notre tour – pour instrumenter un pot de miel sans modifier les logiciels installés sur ce pot de miel. Mais cette méthode est lourde. Non seulement elle ne permet pas d'obtenir des informations avec certitude, mais elle doit être calibrée en fonction de la vitesse de frappe de l'attaquant observé (ne permettant pas d'observation en temps réel). De plus, dans notre cas, nous disposons du contrôle de la machine qui va être ciblée par l'attaquant. Donc, pour observer l'activité des attaquants qui utilisent des connexions chiffrées, nous pouvons modifier légèrement le pot de miel, au lieu d'utiliser cette méthode.

Comme modification, nous pourrions très bien employer la méthode "homme au milieu" (en anglais, *man in the middle*). Cette méthode utilise le concept de "proxy" en insérant un serveur entre le client et le serveur pour intercepter les accès au service `ssh` du serveur[ste01]. Dans notre cas, nous pourrions intercepter les accès au pot de miel, par l'attaquant, afin de récupérer les frappes de touches de cet attaquant. Mais, plûtot que de déployer un tel outil, il pourrait nous suffire de disposer de la clé de session utilisée pendant la communication, $g^{xy} \mod p$. Pour ce faire, une modification minime du serveur `ssh` pourrait suffire. Cette modification consiste à rendre prédictive la génération de la valeur aléatoire y par le serveur. Nous disposerions alors des valeurs p, g et $g^x \mod p$ qui circulent en clair sur le réseau et de la valeur y. Il serait

alors possible de calculer $g^{xy} \mod p$, pour ainsi déchiffrer les échanges entre les attaquants et le serveur `ssh` du pot de miel. Cette approche est moins détectable que la modification des logiciels installés sur le pot de miel mais elle se trouve limitée par un point important. Un attaquant peut installer sur le pot de miel ses propres librairies de chiffrement. Dans ce cas, nos modifications seront inutiles et nous nous trouverons dans l'impossibilité de déchiffrer les connexions. Reconstituer l'activité de l'attaquant sera alors irréalisable. Cette solution n'est donc pas suffisante. Nous devons à nouveau considérer une modification plus importante.

Effectuer une importante modification sur le pot de miel consiste à transformer les logiciels installés. Bien entendu, pour ce faire, nous n'allons pas modifier tous les programmes du système de fichiers en utilisant les techniques de *Hijacking*. Le travail serait trop long et cette stratégie est facile à contourner par le téléchargement sur le pot de miel de nouveaux fichiers exécutables. Nous allons donc adopter la stratégie répandue qui consiste à modifier le noyau du système d'exploitation. Deux approches existent. La première consiste à créer un greffon à appliquer au noyau du système d'exploitation et la deuxième consiste à utiliser un module à insérer dynamiquement dans le noyau. Nous avons opté pour la première approche, afin d'obtenir un bon niveau de transparence. Les implémentations présentées dans la section 3.3 réalisent de telles transformations. Elles emploient des techniques bien connues des attaquants. Elles sont, de ce fait, facilement détectables. De plus, la plupart emploient des modules à charger dynamiquement. Nous allons donc tenter de créer une implémentation originale, sous forme d'un greffon.

Le greffon va opérer à plusieurs niveaux. Il va modifier deux routines du noyau. La première concerne l'exécution des programmes. La seconde concerne la communication avec un utilisateur distant. En plus de ces modifications, il est nécessaire de donner la possibilité à l'administrateur du pot de miel d'enrichir les informations collectées. Cette capacité donne un bon niveau de flexibilité au pot de miel. Elle est réalisée par le greffon en enrichissant le noyau d'un nouvel appel système permettant à des processus exécutés au niveau de l'espace utilisateur de stocker des informations au niveau de l'espace noyau. Des programmes qu'il nous semble pertinent de modifier peuvent profiter de cet appel système. L'un de ces programmes est `ssh`. Nous l'avons modifié de manière à récupérer des informations sur les tentatives de connexions (nom d'utilisateur, mot de passe, adresse de l'attaquant...).

3.4.3 La redirection des rebonds

Nous avons imaginé un mécanisme donnant l'illusion à l'attaquant que ses connexions issues du pot de miel et à destination d'Internet ont réussi. Ce mécanisme réalise des redirections de connexion sélectives et à la volée. Son utilisation est adaptée pour l'observation des attaquants n'ayant aucune connaissance des machines qu'ils attaquent.

Afin de fixer les idées, la figure 3.5 présente une mise en situation de ce principe sur un pot de miel constitué de trois machines, b, c et d. La connexion 1 est initiée par l'attaquant depuis la machine a d'Internet, vers la machine b du pot de miel. Cette connexion permet à l'attaquant de prendre le contrôle de la machine b. La machine b constitue le point d'entrée de l'attaquant dans notre mécanisme. Depuis la machine b, l'attaquant tente d'accéder à la machine e d'Internet en initiant la connexion 2. Cette connexion est bloquée par notre mécanisme. L'attaquant tente alors une autre

connexion, la connexion 3, vers la machine f d'Internet. Cette connexion est acceptée mais redirigée vers la machine c du pot de miel. La redirection donne l'illusion à l'attaquant de contrôler la machine f, mais, en réalité, elle donne le contrôle de la machine c à cet attaquant. En continuant son activité, l'attaquant tente une nouvelle connexion, la connexion 4, depuis la machine b vers la machine g d'Internet. Cette connexion est aussi acceptée, mais, à son tour, redirigée vers la machine d du pot de miel. A présent, depuis la machine c que l'attaquant contrôle depuis la connexion 3, la connexion 5 à destination de la machine g doit aussi être redirigée vers la machine d du pot de miel.

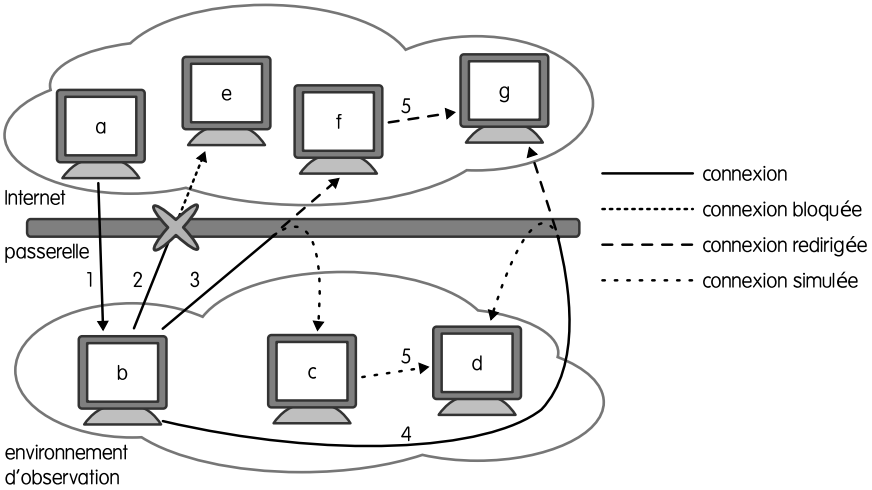


Fig. 3.5 – Le principe de redirection à la volée

Le mécanisme présenté dans cette section permet d'observer l'activité de l'attaquant, sur les différents rebonds qu'il aurait voulu employer pour réaliser l'attaque. L'intérêt majeur est qu'il donne l'illusion à l'attaquant que ses attaques sur des machines d'Internet ont réussi. En revanche, si l'attaquant possède une connaissance des machines sur lesquelles il souhaite rebondir, cette supercherie est identifiable. Par exemple, dans la figure 3.5, nous pouvons supposer que l'attaquant contrôle déjà les machines a, e et f. Il peut alors tester, après la connexion 3, si la machine sur laquelle il est connecté est bien la machine f. Cette limitation existe. Cependant, il est intéressant de valider cette hypothèse et d'observer le comportement d'un attaquant face à ce genre de situation. A l'instar de `HONEYD`, qui nous donne beaucoup d'informations sur le comportement malicieux d'Internet malgré une furtivité limitée, nous pensons que notre système nous permettra tout de même d'en savoir plus. Il nous dira aussi s'il est utile d'augmenter le niveau d'intéraction de la simulation du rebond.

Concernant la conception de ce mécanisme, la modification de la passerelle semble être toute indiquée. Toutes les connexions mettant en jeu le pot de miel transitent par ce nœud du réseau. Un point essentiel à ne pas négliger doit évidemment être la transparence du mécanisme vis-à-vis des attaquants, afin de collecter des données fiables, i.e. non biaisées pour un comportement soupçonneux des attaquants. Pour cela l'implémentation doit respecter trois caractéristiques : la flexibilité, la cohérence et la performance. La flexibilité offre la possibilité à un administrateur d'adapter le

principe de redirection à ses besoins. La cohérence permet, en fonction des besoins de l'administrateur, de cacher au mieux cette "supercherie" aux yeux des attaquants. La performance vise à ne pas trop augmenter la latence des communications afin de ne pas éveiller les soupçons des attaquants.

3.4.4 Architecture générale

Les mécanismes d'observation des connexions `ssh` et de redirection des rebonds fonctionnent indépendamment l'un de l'autre. De plus, ils sont installés en des points différents. Le premier s'installe sur les machines virtuelles du pot de miel et le second s'installe sur une passerelle. Cette dernière peut être confondue avec la machine physique. Effectivement, à l'instar d'une passerelle, la machine physique gère le routage des communications mettant en jeu les machines virtuelles. Plus précisément, nous étendons le pare-feu de la machine physique. La figure 3.6 présente l'architecture générale du pot de miel. La suite de ce chapitre vise à expliquer comment la conception des différents composants a été réalisée.

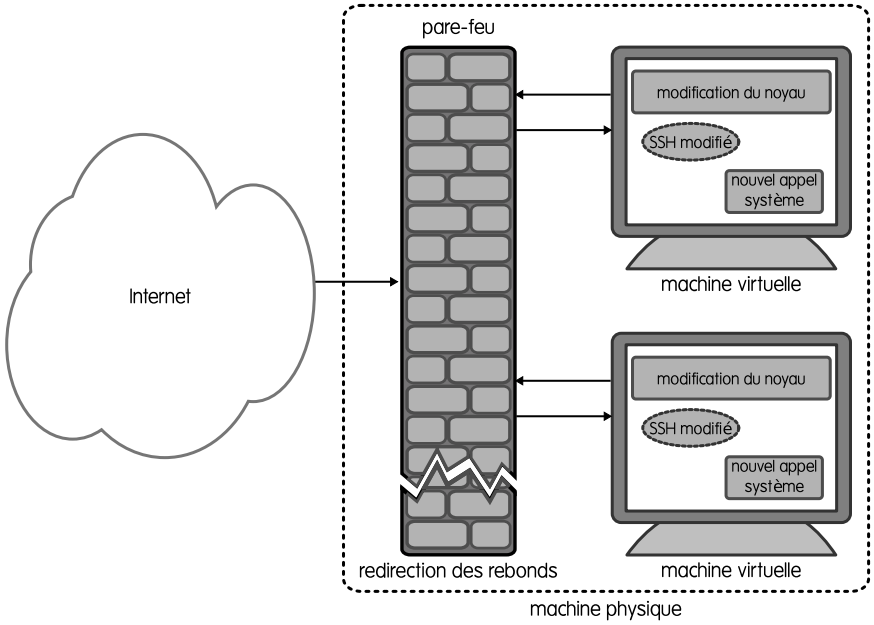


Fig. 3.6 – Architecture générale du pot de miel haute interaction

3.5 Conception et implémentation

Avant de décrire l'implémentation, il convient de choisir les systèmes d'exploitation que nous allons mettre à disposition des attaquants. Les vulnérabilités des systèmes d'exploitation qui ne sont pas couramment utilisés ne sont pas très connues. Nous allons donc choisir des systèmes d'exploitation répandus pour attirer beaucoup d'attaquants. De plus, nous avons besoin de modifier le code source du système d'exploitation. Notre choix s'est donc porté sur les systèmes d'exploitation Gnu/Linux, dont le code source est libre. Ces derniers sont de plus en plus répandus et offrent, à la

base, beaucoup de mécanismes pour communiquer en réseau. Ils ont été installés sur un environnement basé sur VMware. Dans la suite, nous décrivons l'implémentation de notre pot de miel haute interaction, à commencer par la modification des systèmes d'exploitation des machines virtuelles. Ensuite, nous présentons le processus d'archivage des données collectées. Pour finir, nous décrivons les modifications à apporter sur le pare-feu pour mener à bien la redirection des rebonds.

3.5.1 La modification du noyau des systèmes d'exploitation

Dans la section précédente, nous avons choisi d'effectuer trois modifications au niveau des systèmes d'exploitation des machines virtuelles. La première modification intervient au niveau du pilote `tty` permettant au système de dialoguer avec un client distant. Ce pilote `tty` est l'interface entre le système d'exploitation et le terminal de l'utilisateur distant. Il fournit les routines d'écriture et de lecture sur le terminal. Ces routines sont exécutées lors des appels système `read` et `write`, si le fichier sur lequel l'appel système est opéré est un pilote `tty`. La modification de ces routines nous permet de savoir exactement ce que voit et saisit l'utilisateur. Cette modification est réalisée comme indiqué sur la figure 3.1, pour la lecture, en insérant un appel à notre fonction de sauvegarde d'information nommée `log_tty_read`. Nous aurions pu modifier directement les appels système `read` et `write` génériques, mais nous aurions stocké une quantité trop importante d'informations (lecture dans un fichier, lecture d'un fichier exécutable, lecture sur une socket...). Le temps nécessaire au stockage aurait ainsi pu alerter les attaquants. Par conséquent, il aurait été nécessaire de filtrer le surplus d'informations, ce qui est fastidieux à réaliser au niveau des appels système `read` et `write` génériques.

Listing 3.1 – Interception d'un appel système `read` sur un pilote `tty`

```
static ssize_t tty_read(struct file * file, char __user * buf,
                        size_t count, loff_t *ppos)
{
  int i;
  struct tty_struct * tty;
  struct inode *inode;
  struct tty_ldisc *ld;

  ...
  log_tty_read(tty, buf, count); // Interception de la routine.
  return i;
}
```

La seconde modification porte sur la routine d'exécution de programmes. Lors de l'appel système `exec`, effectué entre autres par la fonction `sys_execve` (figure 3.2), un fichier est récupéré, instancié en mémoire en tant que processus et exécuté. En modifiant cet appel système, nous pouvons ainsi déterminer la liste des fichiers exécutés par l'intrus. Cette modification est importante car la modification précédente ne

reflète pas toujours les programmes exécutés par un intrus, notamment si l'exécution de ceux-ci est différée.

Listing 3.2 – Interception de l'appel système `exec`

```
/*
 * sys_execve() executes a new program.
 */
int do_execve(char * filename,
        char __user *__user *argv,
        char __user *__user *envp,
        struct pt_regs * regs)
{
        struct linux_binprm *bprm;
        struct file *file;
        int retval;
        int i;

        log_exec(filename, argv); // Interception de la routine.
        retval = -ENOMEM;
        ...
}
```

La dernière modification consiste à ajouter un nouvel appel système. Tout d'abord, la routine correspondante est créée et ajoutée dans la hiérarchie du code source de LINUX. A ce niveau, la logique de fonctionnement de l'appel système est en place. Seulement, les processus de l'espace utilisateur ne savent pas encore comment demander l'exécution de cet appel système. Pour ce faire, le nom de l'appel système est ajouté dans la table des appels systèmes, elle-même déclarée dans le fichier `syscall_table.S`. L'indice dans cette table représente le numéro de l'appel système. Pour que l'appel système soit connu des programmes de l'espace utilisateur, le numéro correspondant est inscrit dans le fichier `unistd.h`. L'implémentation est effectuée en déclarant une fonction dont le prototype correspond au nom de l'appel système. Pour l'activation de ce nouvel appel système depuis l'espace utilisateur, la macro `_syscall2` a été utilisée. Elle prend en paramètre plusieurs informations, dont le nom de l'appel système. A partir de ce nom, elle récupère le numéro associé, elle initialise les registres du processeur en fonction des autres paramètres et elle demande l'exécution de l'appel système, par une interruption par exemple.

3.5.2 Archivage des données collectées

La récupération des informations pour leur stockage est généralement coûteuse en terme de temps d'exécution. Des attaques statistiques peuvent être pratiquées pour détecter des surcharges dans l'exécution des appels système et, par conséquent, démasquer le pot de miel. Dans [AGJ05], les auteurs ont mis en place un mécanisme permettant de moduler la quantité d'informations récupérée en fonction de la charge du système d'exploitation observé. Pour notre implémentation, cette récupération sera

réalisée de manière périodique et indépendamment de l'activité du système d'exploitation invité. Elle n'aura pas lieu à chaque activité sur le pot de miel. Aussi, qu'il y ait eu une activité sur le système d'exploitation invité ou non, la récupération est différée et effectuée à des heures fixes. De cette manière, nous contrôlons mieux la surcharge d'exécution imputée à nos modifications du noyau. Cette approche rend plus difficile la découverte du pot de miel par des attaques statistiques.

Différer la récupération implique que les informations doivent être emmagasinées temporairement sur le système d'exploitation invité. Les modifications présentées précédemment sont opérées au niveau du noyau. Donc, par simplicité, ces informations seront emmagasinées temporairement dans une zone de la mémoire virtuelle de l'espace noyau avant d'être envoyées dans la base de données.

Cette zone mémoire est statique. Autrement dit, elle ne change pas de place dans la mémoire virtuelle lors d'une exécution du système d'exploitation invité. Elle est précédée d'un motif de caractères unique afin de permettre sa localisation par correspondance de motifs. Nous reviendrons sur l'utilité de la localisation de ce motif plus loin dans cette section.

La zone mémoire possède une taille fixe. Nous n'utilisons pas le mécanisme d'allocation dynamique de mémoire pour l'adapter à la taille des informations. Ce mécanisme peut effectivement surcharger l'exécution des appels système.

En cas d'activité intense sur le pot de miel, la taille de la zone mémoire peut s'avérer insuffisante. Certaines informations risquent d'être perdues. Afin de limiter ces pertes, les informations collectées sont compressées via l'algorithme de compression par dictionnaire `Lzrw1`1[Wil91]. Bien entendu, utiliser un tel algorithme ne permet pas de pallier ce problème, mais présente tout de même l'avantage de permettre d'emmagasiner plus d'informations dans la même zone mémoire. De plus, un algorithme de compression par dictionnaire est très efficace pour compresser des informations présentant beaucoup de redondances. Tel est le cas pour les informations affichées dans un terminal `tty`.

Nous avons limité et non pallié ce problème. Nous ne sommes pas à l'abri des pertes d'informations si l'activité devient vraiment très intense. Il est, au moins, important de détecter ces pertes. Pour cela, nous gérons un compteur au niveau de l'espace noyau. Chaque information à emmagasiner se voit affecter, pour identifiant, la valeur de ce compteur. Le compteur est systématiquement incrémenté, même si la place vient à manquer pour emmagasiner l'information. Ainsi, les pertes seront identifiées par la présence de deux informations emmagasinées consécutivement mais ayant des identifiants non consécutifs.

Les informations sont emmagasinées dans la mémoire de la machine virtuelle. Or, un intrus peut acquérir les privilèges nécessaires pour analyser le contenu de la mémoire virtuelle du système d'exploitation. Identifier la zone contenant les informations emmagasinées devient alors possible. Toutefois, cette identification est rendue difficile par l'emploi de l'algorithme de compression qui brouille un minimum les informations emmagasinées.

3.5.3 Récupération des données archivées

Nous avons présenté la manière dont les informations étaient récupérées et archivées sur le système d'exploitation invité. A présent, nous allons présenter le mécanisme

permettant le renseignement de la base de données.

Pour leur enregistrement dans la base de données, les informations emmagasinées ne sont pas envoyées par le réseau comme dans [AGJ05]. Elles sont récupérées directement par le système d'exploitation hôte : nous profitons du fait que la mémoire du système d'exploitation invité est directement accessible depuis le système d'exploitation hôte. Rappelons que la machine invité s'exécute sur la machine physique. Par conséquent, la mémoire de la machine invité correspond à une partie de la mémoire de la machine physique. De la sorte, il ne nous est pas nécessaire de masquer les interactions entre les deux systèmes d'exploitation. Par contre, cela implique que cette récupération doit être effectuée de manière périodique et indépendamment de l'activité du système d'exploitation invité, comme indiqué précédemment (cf. section 3.5.2). Le système d'exploitation invité n'a pas à avertir qui que ce soit pour indiquer que les informations qu'il contient doivent être récupérées. Quant au système d'exploitation hôte, il effectue la récupération des informations indépendamment de l'activité du système d'exploitation invité (qu'il y ait eu des activités depuis la récupération précédente ou non) et à des heures fixes. Cette récupération consiste à bloquer l'exécution du système d'exploitation invité pendant un cours laps de temps, localiser le motif marquant le début de la zone d'information, récupérer les informations, vider la zone mémoire correspondante et débloquer le système d'exploitation invité. Cette séquence d'opération est réalisée en moins d'une seconde, évitant ainsi de perturber d'éventuels attaquants présents à ce moment. Les informations récupérées sont brutes. Elles sont formatées pour pouvoir être insérées dans la base de données.

3.5.4 Vue globale de la collecte des données

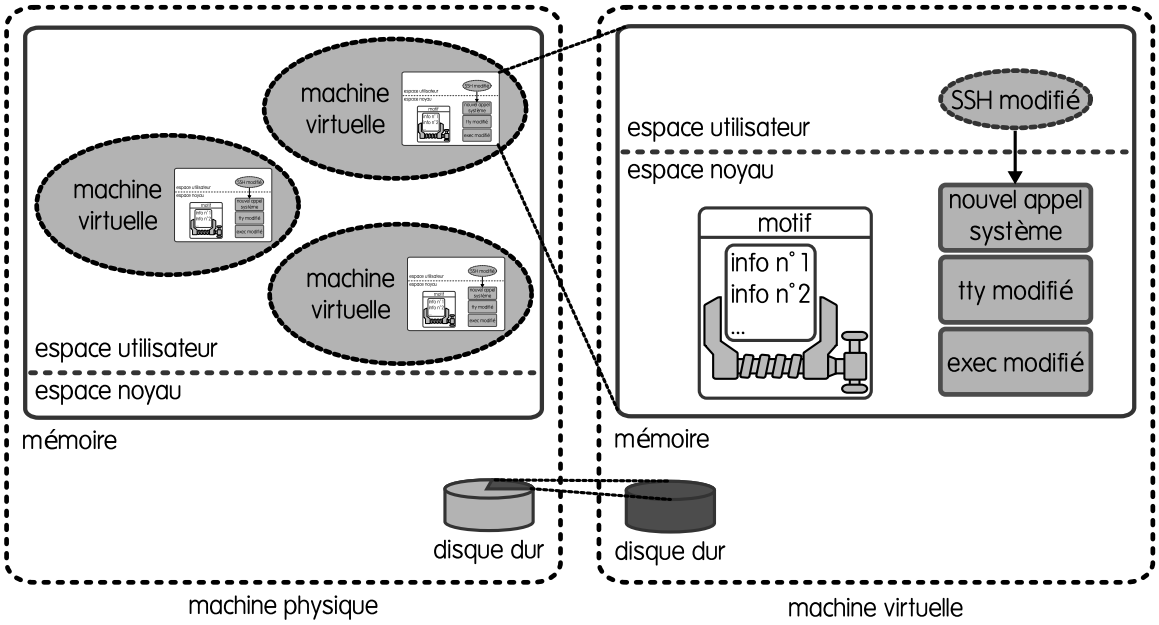


Fig. 3.7 – Implémentation du pot de miel haute interaction

La figure 3.7 présente l'architecture de collecte des informations dans notre pot

de miel haute interaction. Notre implémentation n'est pas réalisée à base de module. Le noyau a été directement modifié. Cette approche confère à notre implémentation une bonne transparence. L'observabilité, quant à elle, est assurée de manière à pouvoir rejouer l'activité de l'intrus. Il nous est aussi possible de savoir quels ont été les programmes exécutés sur le système. Par contre, cette implémentation possède des inconvénients. Tout d'abord, il nous est impossible de détecter l'exécution de programme exploitant la méthode `userland-exec`[rp05][1] (la détection est alors à réaliser à partir des traces réseaux). Ensuite, nous limitons l'observation à des activités `ssh`. L'utilisation par un intrus d'une porte dérobée (en anglais, *backdoor*) permettant le lancement de programme doit faire l'objet d'un traitement supplémentaire, sur la base des traces réseaux (sous réserve que les communications entre la machine de l'intrus et la porte dérobée ne se fassent pas de manière chiffrée).

3.5.5 Le mécanisme de redirection des connexions

La technique de redirection des connexions dans le cadre des pots de miel a fait l'objet de beaucoup de travaux[XJ04, DMC06, BCW+04]. Par exemple, l'architecture `Collapse`, présentée dans [XJ04], intègre un mécanisme pour transférer le trafic malveillant ciblant un système d'un réseau, vers un pot de miel nommé `Collapse Center`. Cependant, ce mécanisme ne traite pas le problème de la redirection du trafic émis depuis le pot de miel et à destination d'Internet. La problématique traitée dans [DMC06] est similaire à la nôtre. Toutefois, la solution proposée n'est pas adaptée à notre objectif. Elle se cantonne à la redirection de connexions basées sur le protocole TCP en utilisant la technique de vol de connexions (en anglais, *hijacking*). De plus, cette solution nécessite la modification de la pile IP du pot de miel. Notre implémentation se destine à être complètement conforme avec le standard IP et, de plus, à être plus générale en acceptant le traitement de divers protocoles : TCP, UDP, ... Nous pouvons aussi mentionner l'architecture hybride de pot de miel, présentée dans [BCW+04]. Le mécanisme de redirection présenté dans ces travaux se destine à l'analyse de malware et de leur propagation. Cependant, peu de détails sont fournis sur la manière dont l'implémentation a été réalisée.

Notre mécanisme de redirection des connexions permet de donner l'illusion à l'attaquant de progresser dans son attaque sur Internet depuis un pot de miel en redirigeant cet attaquant vers un autre pot de miel. Ce mécanisme est présenté, plus en détails, dans [AAN+07a]. Dans la suite, nous en expliquons l'implémentation réalisée sur un système `Gnu/Linux` et doté d'un pare-feu. Pour ce dernier, notre choix s'est porté sur `Netfilter`, pare-feu du noyau installé par défaut sur ces systèmes.

`Netfilter` peut être vu comme un ensemble de 5 chaînes : `INPUT`, `OUTPUT`, `FORWARD`, `PREROUTING` et `POSTROUTING`. Chacune correspond à un point du parcours d'un paquet dans la pile réseau du système d'exploitation. La chaîne qui nous intéresse correspond au traitement d'un paquet juste avant qu'il passe dans l'algorithme de routage : `PREROUTING`. Lui attacher une extension nous permet le confinement de l'activité des attaquants en redirigeant les connexions sortantes vers d'autres machines.

[1] La technique `userland exec` permet la création d'un processus sur un système à partir d'un fichier exécutable disponible sur un autre système. L'exécution se fait via le réseau sans stocker le fichier exécutable sur la machine cible.

Cette extension est implémentée sous la forme d'un module de redirection (cf. figure 3.8). L'enregistrement de ces machines auprès du mécanisme est nécessaire. Il est réalisé depuis l'espace utilisateur pour plus de souplesse. De la sorte, le noyau n'a pas à être modifié pour ajouter une nouvelle machine dans le mécanisme de redirection. De plus, l'utilisateur doit pouvoir configurer le mécanisme de redirection pour ne traiter que certaines connexions. Pour chacune, le choix entre bloquer et rediriger et le choix de la machine vers laquelle rediriger la connexion peuvent être déterministes ou aléatoires. Cette logique de fonctionnement est plus simple à développer au niveau de l'espace utilisateur qu'au niveau de l'espace noyau. L'implémentation est donc répartie entre l'espace noyau et l'espace utilisateur. Le but de la partie de l'implémentation localisée dans l'espace noyau est de déclencher la demande de redirection pour les nouvelles connexions qui nous intéressent. Quant à la partie localisée au niveau de l'espace utilisateur, son but est de déterminer la manière de traiter la demande de redirection. La figure 3.8 présente l'architecture du mécanisme de redirection. Dans la suite de cette section, nous expliquons plus en détails les deux parties constituant le mécanisme.

La partie de l'implémentation localisée au niveau de l'espace utilisateur est composée de deux entités : le *dialog_tracker* et le *dialog_handler*. La première entité joue le rôle d'interface entre notre module de redirection et la logique de fonctionnement. Plus précisément, elle traduit les informations envoyées depuis notre module de redirection pour les rendre compréhensible par la logique de fonctionnement. De cette manière, la portabilité vers d'autres systèmes peut être envisagée. La deuxième entité, le *dialog_handler*, correspond à la logique de fonctionnement. Elle est responsable de la prise de décision du devenir d'une connexion : soit cette connexion est bloquée, soit elle est redirigée. Les machines potentiellement destinataires d'une redirection et des règles de redirection sont enregistrées au niveau de cette entité. Les règles permettent de guider la prise de décision en fournissant la méthode à employer, aléatoire ou déterministe, en fonction des caractéristiques de la connexion. Ainsi, lorsqu'une connexion est initiée depuis le pot de miel à destination d'Internet, notre module de redirection avertit le *dialog_tracker*, qui traduit le message pour le *dialog_handler* qui, à son tour, décide du devenir de cette décision et de la machine vers laquelle rediriger si besoin. Après la prise de décision, le *dialog_handler* informe le *dialog_tracker* du résultat qui, à son tour, informe notre module de redirection.

Aucune recompilation du noyau n'est nécessaire pour le développement du module de redirection. Il joue le rôle d'intermédiaire entre le pare-feu et la partie de l'implémentation localisée au niveau de l'espace utilisateur. Il s'enregistre auprès du pare-feu afin de traiter le premier paquet de chaque connexion. Le suivi de connexion `conntrack` offert par le pare-feu nous assure que les paquets suivants subissent le même traitement que le premier. Le mécanisme de translation d'adresses, `nat`, rend effectif la demande de redirection déclanchée par le module de redirection. Pour les autres connexions, ce module sollicite les entités de l'espace utilisateur, attend la règle associée et traite lui-même la connexion avec cette nouvelle règle. Pour faciliter la communication depuis l'espace noyau vers l'espace utilisateur, nous utilisons la librairie `libnetfilter_queue`. Pour la communication inverse, c'est-à-dire pour l'ajout de nouvelles règles, nous utilisons les sockets `netlink`. A chaque retour d'information depuis le *dialog_tracker*, il mémorise la règle à utiliser. Ainsi, les connexions pour lesquelles il est capable d'associer une règle sont directement traitées par lui-même, sans

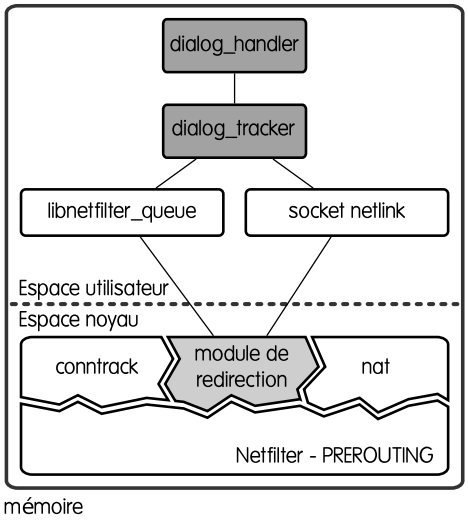


Fig. 3.8 – Architecture du mécanisme de redirection des connexions

solliciter l'espace utilisateur. Les performances sont ainsi améliorées.

3.6 Déploiement

L'implémentation présentée précédemment à permis la mise en place d'un expérimentation au sein du laboratoire. L'objectif est d'analyser le comportement des attaquants qui réussissent à pénétrer le cœur du système. Le moyen qu'ils emploient pour parvenir à y entrer n'est donc pas aussi important que les activités qu'ils réalisent dans ce système. Pour cette raison, nous avons choisi une vulnérabilité simple : des mots de passe simples pour les comptes accessibles par le service `ssh`. Aussi, le pot de miel que nous avons déployé possède un niveau de sécurité moyen. Si le niveau est trop élevé, nous empêcherons l'observation de comportements intéressants. Ensuite, si le niveau de sécurité est trop faible, l'attaquant peut suspecter une anomalie et partir sitôt entré.

Pour le premier déploiement de notre pot de miel, nous avons uniquement mis en place le système de collecte des données, sans activer le mécanisme de redirection. La figure 3.9 présente ce déploiement. Nous avons installé sur le logiciel `VMware` trois machines virtuelles de type `Gnu/Linux` équipées des outils usuels (compilateur, éditeur de texte, ...) et instrumentées avec le mécanisme de collecte de données. Deux de ces machines (M_1 et M_2) sont accessibles depuis Internet. La troisième machine (M_3) est accessible uniquement depuis les deux premières. Seules les connexions entrantes et à destination du service `ssh` sont autorisées par un pare-feu. Quant aux connexions sortantes, elles sont tout simplement bloquées par ce pare-feu. Nous avons déclaré sur les trois machines des comptes avec des mots de passe simples, de telle manière que les attaquants puissent facilement les deviner. Ce premier déploiement date du 6 janvier 2006. Actuellement, nous recensons 419 jours de collecte de données.

Après nous être familiarisés avec les données collectées et les activités des attaquants, nous avons mis en service le mécanisme de redirection. La figure 3.10 présente

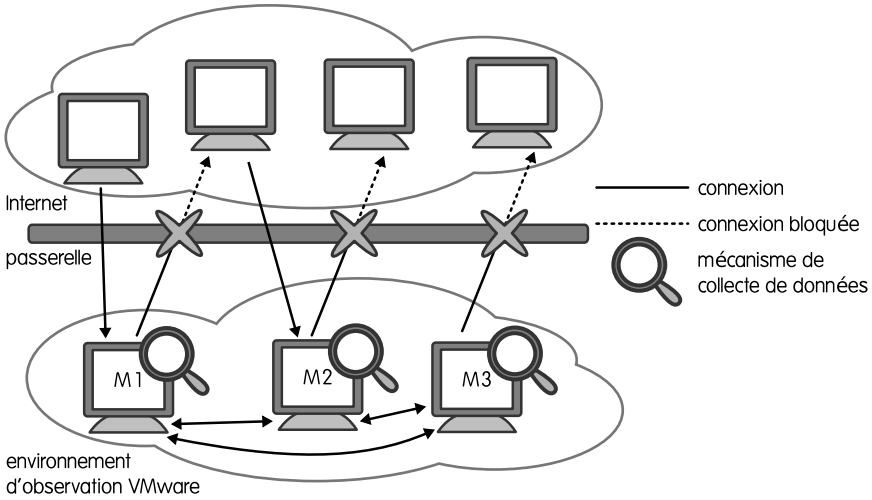


Fig. 3.9 – Configuration du premier déploiement

la configuration de ce second déploiement. A cet effet, nous avons installé, sur la base de l'outil Qemu, cinq machines, équipées de la même manière que pour le premier déploiement. Nous avons opté pour l'utilisation de Qemu afin de faciliter l'automatisation des tâches telles que l'exécution au démarrage de la machine physique, la maintenance à distance, etc. Par rapport au déploiement précédent, deux nouvelles machines ont été ajoutées (R_1 et R_2). Ces deux machines sont uniquement utilisées pour rediriger les attaquants si le mécanisme de redirection accepte les connexions sortantes. Ce deuxième déploiement date du 18 décembre 2006.

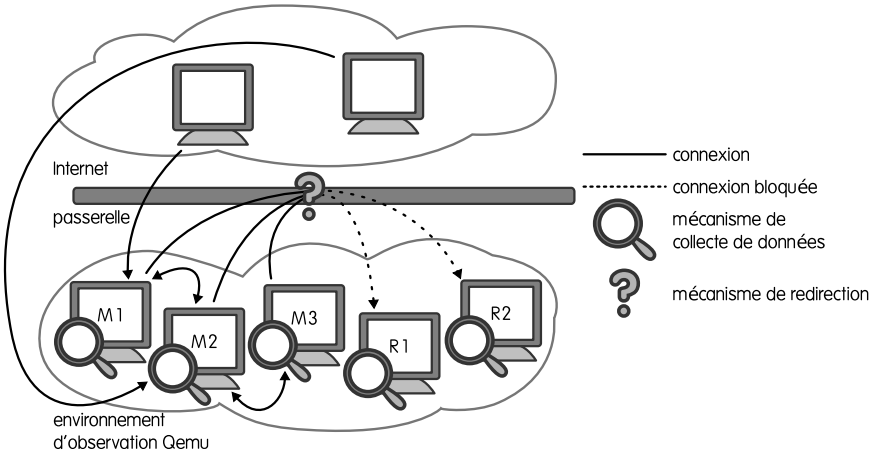


Fig. 3.10 – Configuration du deuxième déploiement

Les données collectées durant ces expérimentations ont été placées dans une base de données Mysql. Pour l'accès à toutes ces informations, une interface en Java a été développée. Cette interface nous facilite l'analyse des données. La suite de ce manuscrit présente les résultats de ces analyses.

3.7 Conclusion

Dans ce chapitre, nous avons défini des caractéristiques des pots de miel haute interaction. Différentes implémentations ont ensuite été étudiées, en considérant ces caractéristiques. L'étude du comportement des attaquants humains est notre objectif. Nous avons identifié l'architecture qui convient pour atteindre cet objectif. Aucune des implémentations proposées dans d'autres travaux n'est adaptée aux objectifs que nous nous sommes fixés. Nous avons alors présenté la conception d'une implémentation correspondant à notre architecture. Cette implémentation a été déployée sur Internet. Elle est active depuis plusieurs mois. Nous avons collecté beaucoup de données. Le chapitre suivant est consacré à l'étude de ces données.

Chapitre 4

Caractérisation des attaques observées sur le pot de miel haute interaction

Les analyses des données collectées à partir des pots de miel haute interaction sont importantes pour comprendre et caractériser les comportements des attaquants une fois introduits dans le système. Ces analyses de nature qualitative permettent de constituer des bases de connaissance dont l'utilité est, entre autres, d'aider les recherches sur un système suite à une intrusion (en anglais, *forensics*) et de guider l'élaboration des modèles permettant l'évaluation des risques liés aux malveillances.

Ce chapitre présente la méthodologie et les résultats de l'analyse des données collectées à partir du pot de miel haute interaction présenté dans le chapitre 3. Cette méthodologie permet de mettre en évidence un processus d'intrusion en deux étapes. La première concerne les connexions permettant à un attaquant d'obtenir un nom d'utilisateur et un mot de passe valides, tous deux nécessaires pour accéder à la machine ciblée. Etudier ces connexions permet de savoir si les attaquants adaptent, en fonction du système ciblé, leurs outils qui permettent d'obtenir ces informations. La deuxième étape concerne les connexions réussies suivies d'éxecution de commandes, permettant à l'attaquant de mener son action. L'étude des actions menées par les attaquants au sein du système peut dévoiler leurs intentions et leurs compétences. Ce processus en deux étapes reflète le comportement de l'attaquant. Plus précisément, ce chapitre vise à étudier ce processus.

Ce chapitre est structuré de la manière suivante. Tout d'abord, nous brossons une description de l'activité `ssh` observée sur le pot de miel. Puis, nous regroupons les connexions au service `ssh` afin d'identifier les différents comportements observés, dans le but de faciliter l'analyse. Ensuite, nous étudions ces comportements afin d'exhiber et d'analyser les deux étapes du processus d'intrusion.

4.1 Aperçu de l'activité des connexions

Une connexion au service `ssh` est l'unité d'observation fournie par le pot de miel. Chaque connexion t est caractérisée par cinq attributs : la date de la connexion $date(t)$, l'adresse à l'origine de la connexion $adresse(t)$, le couple (nom d'utilisateur, mot de

passe) tenté $couple(t)$, un booléen indiquant si ce couple est valide $valide(t)$ et un booléen indiquant si la connexion est suivie d'une saisie de commandes $commande(t)$. Bien entendu, une connexion peut être suivie d'une saisie de commandes uniquement si le couple tenté est valide.

Notre expérimentation s'est déroulée en deux temps. Nous avons commencé par déployer notre architecture avec le service `ssh` correctement configuré, mais sans y avoir déclaré d'utilisateurs. Pendant approximativement le premier mois de l'expérimentation, nous avons observé les connexions à ce service. Toutes ces connexions n'ont pu qu'échouer. Nous en avons donc profité pour identifier les couples les plus tentés. Ce constat nous a permis de déclarer sur le pot de miel des couples qui ont souvent été testés et d'autres qui n'ont jamais été testés. Le nombre de couples créés est 21. Dans la suite, nous présentons des analyses préliminaires sur les connexions et nous en présentons les origines. Pour finir, nous discutons de la méthodologie d'analyse de ces connexions.

4.1.1 Analyses préliminaires

Durant les 419 jours de collecte de données concernant la période du 5 janvier 2006 au 20 mars 2007, nous avons recensé un total de 552 362 connexions au service `ssh`. 299 d'entre elles ont réussi. Ces dernières ne constituent qu'une faible proportion : moins de $0,05\%$. Toutes ces connexions ont été réalisées par 654 adresses différentes.

Classement	Nombre de connexions	nom d'utilisateur	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`
11	660	`sales`	`sales`
12	659	`test`	`123456`
13	650	`web`	`web`
14	597	`ftp`	`ftp`
15	584	`michael`	`michael`
16	574	`paul`	`paul`

FIG. 4.1 – Liste des couples les plus employés

Pour chacune de ces connexions, un couple a été fourni au service `ssh`. Certains de ces couples ont été utilisés plusieurs fois. Nous en dénombrons 98347 différents. Le tableau 4.1 présente les couples les plus employés. La première remarque faite en observant ce tableau concerne les mots de passe employés : ils sont souvent identiques au nom d'utilisateur correspondant. Le fait que ces couples "simples" soient beaucoup

tentés montre que beaucoup d'utilisateurs emploient ce genre de mots de passe, à la place de mots de passe compliqués, plus difficiles à deviner, mais aussi plus difficiles à retenir. Un autre point important que ce tableau met en évidence est l'attrait particulier pour certains noms d'utilisateur. Les noms d'utilisateur `root` et `test` sont beaucoup employés, avec différents mots de passe.

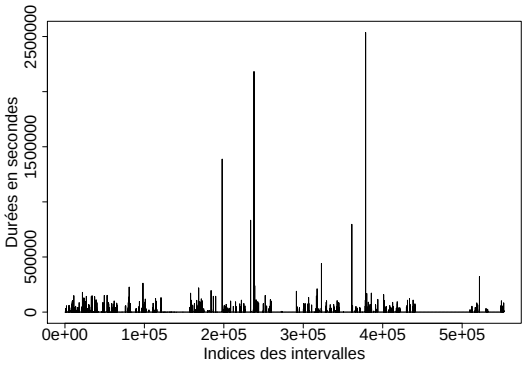


FIG. 4.2 – Évolution de la durée entre les connexions successives

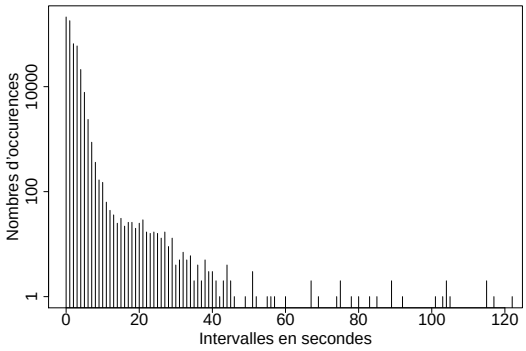


FIG. 4.3 – Fréquences des durées entre connexions successives

Nous disposons des dates auxquelles ces connexions ont été réalisées. En ordonnant ces dates et en calculant les variations, nous obtenons les durées entre deux connexions successives. La figure 4.2 présente l'évolution de ces durées. Nous remarquons sur cette figure des regroupements de durées très courtes, formant des paliers proches de l'axe des abscisses : beaucoup de connexions ont été observées en un laps de temps très court. Certains points correspondent à des durées très longues : aucune connexion n'a été observée pendant la période correspondante. Notamment, le plus grand pic correspond à une coupure intentionnelle du pot de miel, pendant la période des vacances de Noel. Les deux autres grands pics correspondent à des coupures d'électricité. La figure 4.3 présente les fréquences importantes de ces durées. Nous constatons un nombre élevé de durées proches de 0. Les paliers proches de l'axe des abscisses semblent être prédominants, sur la figure 4.2. Le plus long palier regroupe 12939 connexions pour une durée de 25 minutes (l'échelle de l'axe des ordonnées est trop grande pour localiser avec précision ce palier). Autrement dit, lorsque deux connexions sont très proches l'une de l'autre, il y a une forte probabilité pour que la prochaine connexion soit réalisée dans les secondes qui suivent. Ce phénomène a de fortes chances d'être lié à la nature automatique et sagace de certaines activités.

Parmi toutes ces connexions, certaines ont réussi. Nous nous sommes alors concentrés sur la rapidité avec laquelle les couples valides étaient découverts et utilisés. A cet effet, nous avons calculé deux temps. Le premier est celui qui s'est écoulé entre la date de déclaration d'un couple et la date de la première connexion réussie avec ce couple (τ_1). Le deuxième est celui qui s'est écoulé entre la date de cette première connexion réussie et la date de la première connexion réussie avec ce même couple et avec saisie de commande (τ_2). La figure 4.4 présente le lien entre ces deux temps.

Les résultats obtenus sont présentés dans le tableau 4.5. La première colonne identifie le couple déclaré et les deux colonnes suivantes correspondent aux temps présentés précédemment. La seconde colonne indique que les couples que nous avons déclarés, et qui utilisent des mots de passe simples, sont rapidement trouvés par les attaquants. La

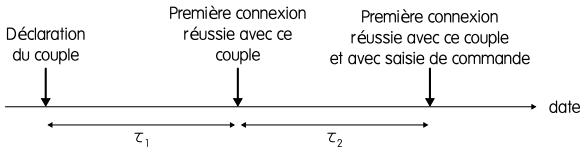


FIG. 4.4 – Lien entre les valeurs τ_1 et τ_2

Couple	τ_1	τ_2
C_1	1 jour	4 jours
C_2	un jour et demi	4 minutes
C_3	15 jours	1 jour
C_4	5 jours	10 jours
C_5	5 jours	0
C_6	1 jour	4 jours
C_7	5 jours	8 jours
C_8	1 jour	9 jours
C_9	1 jour	12 jours
C_{10}	3 jours	2 minutes
C_{11}	7 jours	4 jours
C_{12}	1 jour	8 jours
C_{13}	5 jours	17 jours
C_{14}	5 jours	13 jours
C_{15}	9 jours	7 jours
C_{16}	1 jour	14 jours
C_{17}	1 jour	12 jours

FIG. 4.5 – Temps entre les premières connexions pour chaque couple déclaré

troisième colonne montre qu'il y a un grand écart entre la première connexion réussie et la première connexion réussie avec saisie de commandes. Le compte C_5 constitue un cas particulier : le temps qui s'est écoulé entre la première connexion réussie et la première connexion réussie avec saisie de commandes est nul. Un temps nul signifie que la première connexion réussie est aussi la première connexion réussie avec saisie de commande. L'attaquant s'est connecté directement et a aussitôt saisi des commandes.

4.1.2 Origine des connexions

Certaines connexions ont été suivies d'une saisie de commandes. Ce sont des *connexions d'action*. D'autres connexions, qui constituent la majorité, n'ont pas donné lieu à une saisie de commande. Ce sont des *connexions d'authentification*, qui peuvent avoir été générées pour des raisons diverses[ANK+06, RBC07, Sei].

La première raison correspond à une erreur liée à l'utilisateur. Un utilisateur peut se tromper dans la saisie de l'adresse de la machine à laquelle il tente de se connecter. Après quelques tentatives, voire une seule, la personne se rend vite compte de son erreur. Pour peu que l'adresse incorrecte corresponde à une des adresses de notre pot

de miel, nous observons quelques connexions d'authentification isolées, qui ont échoué.

La deuxième raison est relative à une activité malveillante nommée *attaque par dictionnaire*. Du point de vue du pot de miel, une attaque par dictionnaire correspond à une succession de connexions d'authentification, dans un laps de temps très court. Le but recherché est de deviner un ou plusieurs couples valides. Lors d'une attaque par dictionnaire, la plupart des connexions d'authentification échoue. Certaines peuvent réussir, marquant alors la découverte de couples valides.

Un attaquant ne lance pas une attaque par dictionnaire uniquement pour trouver un couple valide. Il cherche surtout à exploiter les ressources de cette machine pour poursuivre son activité. Cette remarque nous amène à la dernière raison à ces connexions. Elle correspond à l'*intrusion* à proprement parler. Une intrusion est une succession de connexions, dont certaines ont non seulement réussi mais ont aussi permis à l'attaquant de réaliser une action au sein du système ciblé. Autrement dit, une intrusion est composée des deux types de connexions : les connexions d'authentification et les connexions d'action. Toutes les intrusions ont un but particulier, tel que l'identification de machines pouvant être compromises pour faire partie par exemple d'un *botnet*. Les attaquants qui réalisent une intrusion seront nommés par la suite *intrus*.

Dans la suite, nous discutons de la manière d'identifier ces différentes activités.

4.1.3 Discussion et méthodologie

Nous disposons d'un ensemble considérable de données. Nous avons besoin de choisir un bon niveau d'abstraction pour extraire des informations pertinentes caractérisant les attaques observées. Notre but est d'identifier aussi précisément que possible le comportement d'un attaquant sur notre pot de miel. A cette fin, nous allons définir la notion de *session*. Une session est une séquence de connexions proches dans le temps, réalisées par une même adresse et à destination de notre pot de miel. Nous en donnerons une définition formelle plus loin dans ce chapitre. Intuitivement, cette notion de session se veut représenter le comportement d'un attaquant sur notre pot de miel à un instant donné. Si le même attaquant réalise une séquence de connexions proches le jour J, qu'il ne fait plus aucune connexion ensuite pendant 2 jours et qu'il réalise à nouveau une séquence de connexions proches le jour $J+3$ par exemple, nous aurons donc identifié deux sessions du même attaquant. L'ensemble de ces sessions représente le comportement de l'attaquant sur notre pot de miel.

Une session peut être une attaque par dictionnaire, une intrusion ou une attaque d'une autre nature. Autrement dit, une session peut être constituée de connexions d'authentification uniquement ou de connexions d'actions uniquement mais aussi de connexions d'authentification et de connexions d'action (dans le cas par exemple où l'attaquant se trompe une ou deux fois de mots de passe avant de pénétrer sur le système). Il nous faut donc essayer d'extraire les sessions de toutes les connexions au service `ssh` que nous avons enregistrées.

4.2 Construction de l'ensemble des sessions

L'identification des sessions doit obéir à deux contraintes, spatiale et temporelle, pour assurer la cohérence. De manière pratique, le temps entre les connexions et les

adresses sources de ces connexions peuvent jouer le rôle d'indicateur pour identifier les sessions.

Nous proposons donc d'utiliser une démarche composée de deux étapes. La première étape consiste à regrouper les connexions en fonction de deux critères, spatial et temporel, que nous allons fixer. La deuxième étape consiste à filtrer ces regroupements pour identifier les différents types d'attaque. Dans la suite, nous présentons plus en détails la manière dont sont identifiées ces sessions.

Dans cette section, nous présentons la méthode de regroupement basée sur l'algorithme de *fenêtre glissante* avec pour paramètre un seuil. Nous illustrons ensuite cet algorithme sur un exemple. Pour finir, nous discutons le problème du choix de ce seuil et appliquons l'algorithme aux données.

4.2.1 Fenêtre glissante

La méthode de "fenêtre glissante" utilise un seuil de densité minimal pour effectuer une classification des éléments. Afin d'expliquer le seuil de densité minimal, nous allons faire une analogie avec les images que nous visualisons tous les jours. Sans forcément s'en rendre compte, les posters, affiches et prospectus, que nous pouvons rencontrer au cinéma ou dans la rue, sont du type "pointilliste". Cette caractéristique est principalement due aux méthodes d'impression employées. La plupart des imprimantes utilisées aujourd'hui sont quadri-chromatiques. Elles fonctionnent avec quatre couleurs primaires. Pour restituer une couleur, elles déposent, sur le support d'impression, quatre points de couleurs primaires. La taille de chacun de ces points est proportionnelle à la quantité de la couleur primaire correspondante dans la couleur à restituer. Il en résulte que les images sont "tramées". A distance, l'oeil ne perçoit pas ce caractère pointilliste, la trame.

La figure 4.6 illustre cet effet d'optique. Elle représente une feuille sur laquelle ont été déposés des points de couleur noire. En la regardant de près, on observe des points disposés chaotiquement, aléatoirement : la trame. A distance, l'oeil va masquer la localisation de ces points par un effet de moyennage. Il identifie alors trois formes ayant des densités homogènes de points.

La méthode de fenêtre glissante s'appuie sur la notion de seuil de densité minimal pour constituer des groupes, à partir d'un seuil donné et d'un ensemble d'éléments ordonnés. Le seuil représente la limite à ne pas franchir pour que deux éléments consécutifs appartiennent au même groupe. Par cette méthode, nous devons obtenir un ensemble de groupes tel que chaque élément appartient à un groupe et un seul, tel que le temps qui s'écoule entre deux éléments consécutifs d'un même groupe est inférieur au seuil donné et tel que le temps qui s'écoule entre deux éléments appartenant à des groupes différents est supérieur au seuil donné.

Afin de formaliser la méthode de fenêtre glissante, considérons un ensemble d'éléments E. Chaque élément e de cet ensemble est daté : $date(e)$. Les éléments de l'ensemble E sont donc ordonnés par date. Le seuil donné est s. Soit $e_i \mathcal{R}_{(E,s)} e_j$ la relation réflexive et symétrique de proximité conditionnée par le seuil s.

$$\forall (e_i, e_j) \in E^2, \qquad e_i \mathcal{R}_{(E,s)} e_j \Leftrightarrow |date(e_i) - date(e_j)| < s \tag{4.1}$$

Les éléments de l'ensemble E peuvent être représentés dans un graphe non orienté, $G = (X, V)$. L'ensemble des sommets de ce graphe est l'ensemble des éléments, $X =$

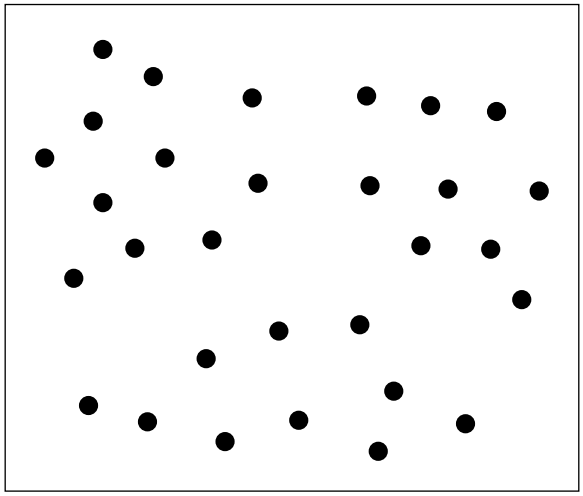


FIG. 4.6 – Impression sur une feuille.

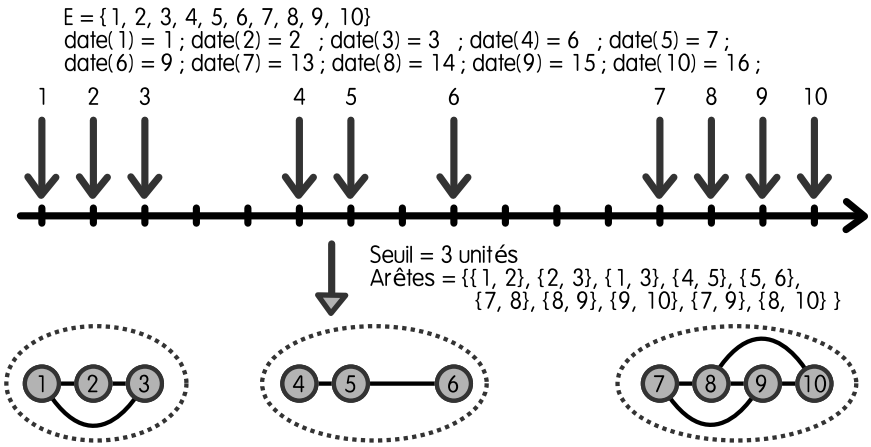


FIG. 4.7 – Exemple d'application de l'algorithme de la fenêtre glissante

E. L'ensemble des arêtes de ce graphe est l'ensemble des couples d'éléments liés par la relation de proximité, $V = e_i, e_j \in E^2 / e_i \mathcal{R}_{(E,s)} e_j$. La figure 4.7 présente un exemple avec 10 éléments, numérotés de 1 à 10. Avec un seuil de 3 unités, le graphe obtenu contient 10 sommets et 10 arêtes. Pour cet exemple, il n'existe pas d'arêtes entre les sommets 3 et 4 et entre les sommets 6 et 7 car le temps qui les sépare est égal ou supérieur au seuil de 3 unités. Pour la même raison, il n'existe pas d'arête entre les sommets 4 et 6 et entre les sommets 7 et 10. Pour finir, on constate des arêtes qui sautent des éléments : entre les sommets 1 et 3, entre les sommets 7 et 9 et entre les sommets 8 et 10.

Algorithme 3 Algorithme de la fenêtre glissante

Entrées: $E, s \in \mathbb{R}^+$
Sorties: $\mathcal{F}_g(E, s) = p \in \mathcal{P}(E)$

```
p ← ∅
- - L'ensemble E ne doit pas être vide.
si E ≠ ∅ alors
  - - Amorcer avec un groupe contenant le plus petit élément, en fonction de la date.
  e ← x ∈ E/∀y ∈ E, date(x) ≤ date(y)
  o ← {e}
  E ← E − o
  - - m contient la date du dernier élément absorbé dans o.
  m ← date(e)
  - - Continuer tant qu'il reste des éléments orphelins.
  tant que E ≠ ∅ faire
    - - Récupération du prochain élément le plus proche, dans E.
    e ← x ∈ E/∀y ∈ E, date(x) ≤ date(y)
    E ← E − {e}
    - - Clôture du groupe si l'écart est trop important.
    si date(e) − m ≥ s alors
      p ← p ∪ {o}
      o ← ∅
    finsi
    - - Grandir le groupe.
    o ← o ∪ {e}
    m ← date(e)
  fin tant que
  - - Ne pas oublier le dernier groupe en cours d'identification.
  p ← p ∪ {o}
finsi
```

Une chaîne du graphe est une suite de sommets du graphe $c = (c_1, ..., c_n)$ telle que deux sommets successifs de cette suite soient liés par une arête. Nous noterons C_G l'ensemble des chaînes du graphe $G = (X, V)$. Une composante connexe d'un graphe est constituée des sommets induits par la relation "est accessible à partir de". L'accessibilité est permise par le biais des arêtes. L'ensemble des composantes connexes d'un graphe forme une partition des sommets de ce graphe. Or, dans notre cas, la présence d'une arête indique une durée entre éléments inférieure au seuil donné s. Donc, une chaîne représente une succession d'écarts inférieurs au seuil. L'ensemble des groupes à obtenir est alors constitué de l'ensemble des composantes connexes du graphe précédent. Pour notre exemple précédent, nous obtenons un ensemble de trois groupes, identifiés sur la figure par des pointillés. Une définition formelle de la fonction permettant d'obtenir l'ensemble des groupes, $\mathcal{F}_g(E, s)$, est donnée dans la suite.

$$C_G = \cup_{n=1}^{\infty}\{c = (c_1, ..., c_n) \in X^n / \forall i < n, \{c_i, c_{i+1}\} \in V\} \tag{4.2}$$

$$\mathcal{F}_g(E, s) = p \in \mathcal{P}(E) / \forall r \in p, \forall (e_i, e_j) \in r^2, \quad e_i \neq e_j \Rightarrow \exists (e_i, ..., e_j) \in C_G \tag{4.3}$$
$$\forall (r, r') \in p^2, \forall (e_i, e_j) \in (r, r'), \quad r \neq r' \Rightarrow \nexists (e_i, ..., e_j) \in C_G$$

L'implémentation de cette fonction repose sur les étapes suivantes (cf. l'algorithme 3). On part d'un groupe contenant le plus petit élément, en fonction de la date. On fait grandir le groupe en absorbant le prochain élément le plus proche, si l'écart entre le dernier élément absorbé et ce prochain élément est inférieur au seuil. Si cet écart est supérieur au seuil, on clôt le groupe et on recommence avec un groupe contenant l'élément qui n'a pas pu être absorbé. On poursuit tant qu'il reste des éléments orphelins. A présent, formalisons la définition d'une session.

4.2.2 Définition formelle d'une session

Soient $\mathcal{T}$ l'ensemble des connexions au service `ssh` observées sur le pot de miel et $A(\mathcal{T})$ l'ensemble des adresses différentes. L'ensemble des sessions $\mathcal{S}(\mathcal{T}, s)$, pour un seuil s et un ensemble de connexions $\mathcal{T}$, est défini comme suit :

$$A(\mathcal{T}) = \{adresse(t)/t \in \mathcal{T}\} \tag{4.4}$$

$$\mathcal{S}(\mathcal{T}, s) = \bigcup_{a \in A(\mathcal{T})} \mathcal{F}_g(\{t \in \mathcal{T}/adresse(t) = a\}, s) \tag{4.5}$$

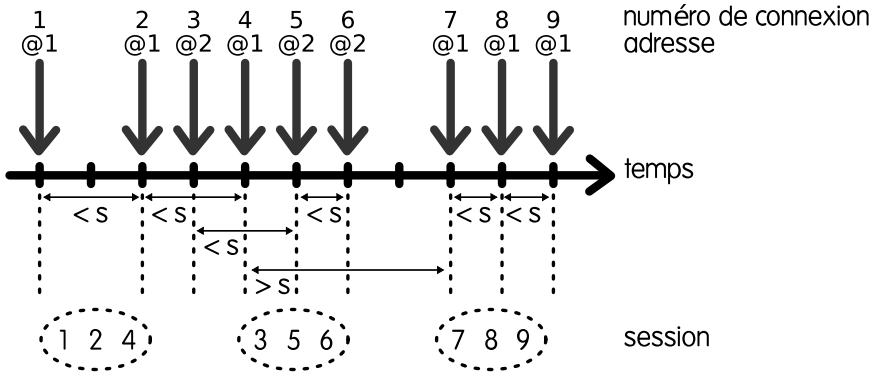


FIG. 4.8 – Exemple de construction d'ensemble de sessions

Nous illustrons cette définition par l'exemple de la figure 4.8. Dans cet exemple, nous considérons un ensemble $\mathcal{T}$ de 9 connexions et un seuil s égal à 3. Le partitionnement de cet ensemble en sessions aboutit à $\mathcal{S}(\mathcal{T}, s) = \{\{1, 2, 4\}, \{3, 5, 6\}, \{7, 8, 9\}\}$. Effectivement, les connexions 1, 2 et 4 sont réalisées par la même adresse $@_1$. De plus, les durées entre les couples de connexions $(1, 2)$ et $(2, 4)$ sont inférieures au seuil s. Concernant le couple de connexions $(1, 4)$, il existe la connexion 2 telle que $1\mathcal{R}_{(\mathcal{T},s)}2 \wedge 2\mathcal{R}_{(\mathcal{T},s)}4$. Le même raisonnement peut être réalisé sur les ensembles $\{3, 5, 6\}$ et $\{7, 8, 9\}$. Aucun des trois ensembles obtenus ne peut être regroupé avec un autre ensemble. Le cas de l'ensemble $\{3, 5, 6\}$ est le plus simple : l'adresse ayant réalisé ces connexions, $@_2$, est différente de l'adresse ayant réalisé les autres connexions, $@_1$. Quant à l'ensemble $\{7, 8, 9\}$, il ne peut pas être regroupé avec l'ensemble $\{1, 2, 4\}$ bien que toutes ces connexions aient été réalisées par la même adresse. Effectivement, les durées entre les couples $(1, 7)$, $(1, 8)$, $(1, 9)$, $(2, 7)$, $(2, 8)$, $(2, 9)$, $(4, 7)$, $(4, 8)$ et $(4, 9)$ sont toutes supérieures au seuil s.

La définition d'une session étant posée, nous présentons l'algorithme permettant de les obtenir, à partir des données (cf. algorithme 4). Pour cet algorithme, nous utilisons la technique de la fenêtre glissante. Son implémentation a été effectuée dans le langage `Perl`.

Algorithme 4 Algorithme de construction de l'ensemble des sessions

Entrées: $\mathcal{T}, s \in \mathbb{R}^+$
Sorties: $\mathcal{S}(\mathcal{T}, s) = f$
  - - *Listes des connexions.*
  $f \leftarrow \emptyset$
  - - *Liste des différentes adresses.*
  $a \leftarrow \{adresse(t)/t \in \mathcal{T}\}$
  pour $i \in a$ **faire**
    - - *Récupération des sessions réalisées par cette adresse.*
    $f \leftarrow f \cup \mathcal{F}_g(\{t \in \mathcal{T}/adresse(t) = a\}, s)$
  fin pour

4.2.3 Choix de la valeur du seuil

A présent, se pose le problème de définir précisément la valeur du seuil s. Ce n'est pas une connaissance a priori des outils d'attaque qui doit nous permettre de déterminer la valeur du seuil s. Cette démarche pourrait nous amener à une situation de sur-apprentissage. C'est une étude des données qui doit nous guider dans le choix de ce seuil. Nous admettons que, lors d'une session, des connexions en rafale peuvent arriver. Du coup, on rejette une analyse fréquentielle. Cependant, on cherche à déterminer le seuil en terme de distance en temps entre des connexions au delà duquel agréger des séquences de connexions revient à agréger des sessions différentes. Autrement dit, nous supposons qu'une session est une région dense séparée par des régions qui le sont moins. Nous sommes intéressés par la plus grande valeur s telle que le risque de regrouper au sein d'une même session des connexions issues de comportements différents est négligeable.

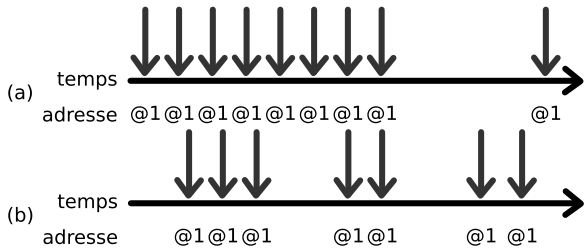


FIG. 4.9 – Cas de figure différents pour la détermination du seuil de regroupement.

Nous avons identifié deux cas de figure, présentés dans la figure 4.9. Dans le premier cas, 8 connexions arrivent en rafale, suivies par une connexion éloignée dans le temps. La fenêtre temporelle contenant les connexions en rafale correspond à une région dense et la fenêtre temporelle contenant la connexion isolée correspond à une région moins dense. Nous ne voulons pas agréger ces deux régions car elles semblent identifier deux

activités différentes. Dans le second cas, nous observons 3 ensembles rapprochés de connexions qui arrivent en rafale. Ces trois ensembles sont équivalents en terme de nombre de connexions. Chacune correspond à une région dense et elle sont proches. Il semble alors judicieux de les agréger car il y a de fortes chances pour que toutes ces connexions fassent partie de la même attaque. Nous allons chercher à agréger ensemble le plus grand nombre de connexions qui correspondent vraisemblablement au même comportement. Nous devons déterminer le plus grand seuil à considérer.

La figure 4.10 présente l'évolution du nombre de sessions en fonction du seuil (exprimé en secondes), en considérant toutes les données recueillies pendant les 419 jours de collecte. Pour un seuil de 0 seconde, aucun regroupement n'est possible. Il y a autant de sessions que de connexions. Le nombre de sessions diminue au fur et à mesure que le seuil augmente. Pour les petites valeurs du seuil, les connexions très proches les unes des autres sont regroupées au sein d'une même session. Pour les grandes valeurs du seuil, les connexions vont très vite être regroupées au sein d'une seule et même session. Pour un seuil égal à la durée de l'expérimentation, il y a autant de sessions que d'adresses différentes.

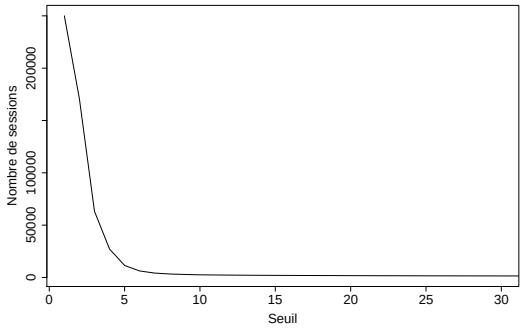


FIG. 4.10 – Evolution du nombre de sessions en fonction du seuil

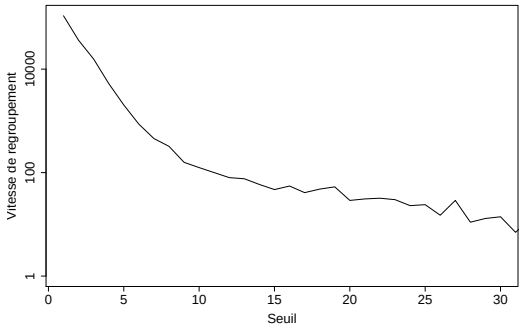


FIG. 4.11 – Vitesse de regroupement des sessions

Nous définissons la *vitesse de regroupement*, $v_r(\mathcal{T}, s)$ comme étant la dérivée de l'évolution du nombre de regroupements en fonction du seuil. Pour déterminer le seuil, nous avons défini un critère subjectif purement qualitatif qui nous indique si un seuil donné s est satisfaisant. Pour le seuil s, ce critère indique si le seuil plus grand d'une seconde, $s + 1$, risque de regrouper des sessions différentes. Pour ce faire, nous nous basons sur la vitesse de regroupement. Elle doit diminuer au fur et à mesure que le seuil augmente. Si elle augmente, alors nous avons franchi une limite pour laquelle nous avons regroupé brutalement plusieurs sessions différentes ensemble. Autrement dit, si elle augmente brutalement, nous regroupons beaucoup de régions denses avec des régions moins denses. Le plus petit seuil cherché, noté $c(\mathcal{T}, s)$, est défini comme suit :

$$v_r(\mathcal{T}, s) = |\mathcal{S}(\mathcal{T}, s) - \mathcal{S}(\mathcal{T}, s + 1)| \tag{4.6}$$

$$c(\mathcal{T}, s) = \max\{s / v_r(\mathcal{T}, s) > v_r(\mathcal{T}, s + 1)\} \tag{4.7}$$

La vitesse de regroupement en fonction du seuil est représentée à la figure 4.11. L'application de ce critère aux sessions nous indique que la valeur 16 est le candidat

idéal pour jouer le rôle de seuil. Effectivement, la vitesse de regroupement entre les seuils 16 et 17 est supérieure à la vitesse de regroupement entre les seuils 15 et 16 : la courbe présente un minimum local. Le nombre de sessions identifiées s'élève à 1940. Cet exemble regroupe plusieurs comportements différents. La suite consiste à identifier des classes de comportement.

4.3 Identification des classes de comportement

Les sessions à notre disposition sont nombreuses et diverses. Nous devons identifier des classes de comportement pour mieux caractériser les processus d'attaque observés. Le but de cette section est d'identifier ces classes de comportement. Une classe de comportement regroupe un ensemble de sessions similaires. Cette classification nous permettra, par la suite, d'étudier séparément les différentes classes qui ont attiré notre attention.

Pour effectuer un regroupement, des critères sont nécessaires. Pour ce faire, nous considérons deux critères pratiques : la *dangerosité* et la *sagacité*.

Une session est dangereuse si l'attaquant à l'origine a réussi à pénétrer le cœur du système et a saisi des commandes : les dégats peuvent être importants et il peut être trop tard. Autrement dit, une session est dangereuse si elle est composée, entre autres, de connexions d'action. Ce critère permet de définir le prédicat $dangereuse(e)$ qui retourne "vrai" si la session e est dangereuse, "faux" sinon.

Une session fait preuve de sagacité si la somme des informations acquises sur le pot de miel n'est pas nulle, à l'issue de l'attaque. Lorsqu'un attaquant teste un couple sur le service `ssh` d'une machine, il obtient plus d'informations si ce couple est valide que s'il ne l'est pas. Bien entendu, savoir qu'un couple n'est pas valide apporte tout de même de l'information à l'attaquant. Donc, nous supposons qu'une session fait preuve de sagacité si le nombre de connexions d'authentification est supérieur à un seuil prédéfini ou si le nombre de connexions d'authentification réussies n'est pas nul. Pour cette étude, nous avons considéré la valeur 9 pour ce seuil, car elle nous semble suffisamment élevée pour éviter de confondre les erreurs des utilisateurs liées à des fautes de frappe avec les attaques par dictionnaire. Ce critère permet, à son tour, de définir le prédicat $sagace(e)$, qui retourne "vrai" si la session e fait preuve de sagacité, "faux" sinon. Ces deux prédicats sont formalisés dans la suite, où $commande(t)$ signifie que la connexion t est une connexion d'action et $valide(t)$ signifie que la connexion inclut une authentification réussie avec un nom d'utilisateur et un mot de passe valides :

$$dangereuse(e) = \begin{cases} vrai, & \text{si} \quad \exists t \in e/commande(t) \\ faux, & \text{sinon} \end{cases} \tag{4.8}$$

$$sagace(e) = \begin{cases} vrai, & \text{si} \quad |e| \geq 9 \cup \exists t \in e/valide(t) \\ faux, & \text{sinon} \end{cases} \tag{4.9}$$

Prenons le premier critère. En l'appliquant sur l'ensemble des sessions, nous identifions deux classes : une première classe des sessions dangereuses de taille 210 et une classe des sessions non dangereuses "a priori", de taille 1730. Cette dernière classe

constitue la principale activité observée sur le pot de miel. L'étude de la répartition des sessions non dangereuses vis-à-vis du second critère, à savoir la sagacité, permet d'identifier 1391 comportements non dangereux mais sagaces et 339 comportements non dangereux et non sagaces. Chacune des 1391 sessions est constituée de plusieurs connexions très proches et correspondant à des couples différents. Certaines ont permis de collecter des informations sur les couples accessibles depuis l'extérieur. Ces sessions s'apparentent à des attaques par dictionnaire, précédant un processus d'intrusion. L'ensemble des attaques par dictionnaire $Dico(\mathcal{T}, s)$ est défini de la manière suivante :

$$Dico(\mathcal{T}, s) = \{e \in \mathcal{S}(\mathcal{T}, s) / \overline{dangereuse(e)} \wedge sagace(e)\} \tag{4.10}$$

Quant aux comportements dangereux, ils correspondent à un attaquant qui est entré au cœur du système pour y lancer des commandes. Ces comportements se nomment intrusions. Le caractère sagace ne nous intéresse pas pour raffiner cette classe. Effectivement, ce n'est pas en comptant le nombre de connexions d'une session que nous pourrons apprécier la dangerosité d'une attaque : une connexion réussie, avec une exploitation d'une vulnérabilité offrant un accès administrateur à l'attaquant est plus dangereuse que plusieurs connexions réussies, avec uniquement des tentatives de rapatriement de fichiers depuis Internet. L'ensemble des intrusions $Intr(\mathcal{T}, s)$ est défini de la manière suivante :

$$Intr(\mathcal{T}, s) = \{e \in \mathcal{S}(\mathcal{T}, s) / dangereuse(e)\} \tag{4.11}$$

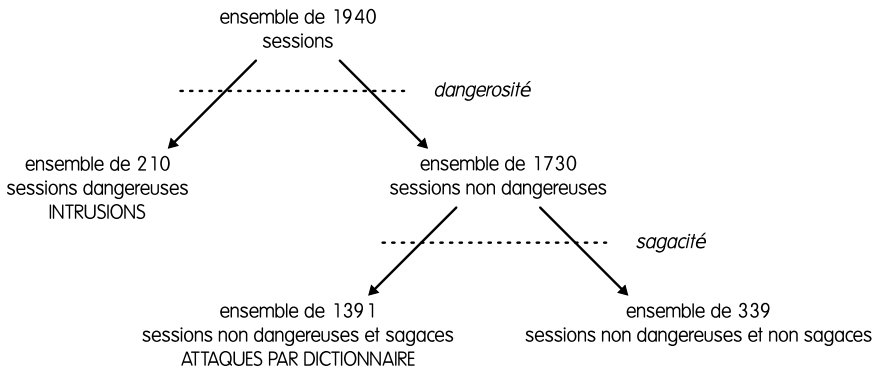


Fig. 4.12 – Classification des sessions

En utilisant deux critères (dangerosité et sagacité), nous avons identifié deux classes de comportements principales. La figure 4.12 représente le dendrogramme associé à cette classification. Toutefois, comme toute méthode de classification, des faux positifs et des faux négatifs peuvent perturber les analyses. Dans notre cas, un faux négatif peut être un exemple de comportement ayant réalisé une attaque par dictionnaire mais ayant collecté peu d'informations. Le critère de sagacité va masquer cet exemple de comportement. En soit, l'étude d'un tel comportement ne nous intéresse pas : a l'issue de cette activité, la connaissance de l'attaquant sur le pot de miel n'a pas évolué. Quant à un faux positif, il peut être un exemple de comportement lié à une erreur d'un utilisateur non malveillant. En se trompant d'adresse de destination,

cet utilisateur peut s'obstiner à établir une communication. Son intention n'est pas mauvaise, mais, du point de vue d'un administrateur réseau, ce comportement se mêle aux comportements agressifs qui doivent être analysés. Nous les assimilons donc à des exemples de comportement malveillants.

Nous ne sommes pas intéressés par les attaques non dangereuses et non sagaces. En revanche, nous nous focalisons sur les comportements dangereux, qui constituent l'objectif de ce chapitre. De plus, nous analyserons aussi les comportements non dangereux mais sagaces. Ils constituent les prémices d'un scénario d'attaque. Etant donné que ces comportements sont longs en terme de durée, leur étude permettrait de les identifier au plus tôt. De la sorte, nous pourrions les bloquer avant leur terme et avant qu'ils n'acquièrent le caractère dangereux. Ces analyses sont réalisées dans la suite de ce chapitre, à commencer par les attaques par dictionnaire.

4.4 Le processus d'attaque

Nous venons d'identifier les différents types d'attaques que nous avons observés sur le pot de miel haute interaction. Chaque type d'attaque repose sur une connaissance à priori du système ciblé par l'attaquant, et permet à cet attaquant de renforcer sa connaissance sur ce système. Par exemple, avant de réaliser une attaque par dictionnaire, l'attaquant a eu connaissance, par un moyen ou par un autre, de le présence du service de connexion à distance sur la machine qu'il cible. De plus, une attaque par dictionnaire lui permet de connaître une partie de l'ensemble des couples (nom d'utilisateur, mot de passe) valides et une partie de l'ensemble des couples non valides sur ce système. Il est alors intéressant d'étudier la manière dont les différents types d'attaques s'agencent pour produire des scénario complexes que nous appellerons *processus d'attaque*, dans le but de mieux comprendre l'organisation des attaquants sur Internet. Cette étude prend en compte l'aspect spacial (nous nous intéressons aux adresses IP ayant réalisées les attaques) et l'aspect temporel (les attaques sont ordonnées dans le temps).

Tout d'abord, nous avons constaté qu'une attaque par dictionnaire ayant permi de découvrir un couple valide précède systématiquement une intrusion réalisée avec ce même couple. Une connaissance à priori du système ciblé est donc requise par un attaquant pour réaliser une intrusion. Elle peut être fournie par une attaque par dictionnaire. De plus, nous avons confronté l'ensemble des adresses ayant réalisé des attaques par dictionnaire à l'ensemble des adresses ayant réalisé des intrusions. L'intersection de ces deux ensembles est vide. Autrement dit, les adresses ayant réalisé des attaques par dictionnaire (resp. intrusions) sont spécialisées dans ce type d'attaque. C'est constatation est d'autant plus intéressante qu'elle est vérifiée sur une longue période de temps (419 jours).

Ensuite, nous avons confronté les adresses ayant réalisées les attaques par dictionnaire ou les intrusions aux adresses observées sur les pots de miel basse interaction, en considérant l'intégralité de la période de collecte (4 ans de collecte). Aucune de ces adresses n'a été observées sur les pots de miel basse interaction. Cette constatation appuye donc la spécialisation des adresses à un type d'attaque. De plus, il est peu probable qu'un attaquant exécute une attaque par dictionnaire sur une machine cible sans se douter à priori de la disponibilité du service de connexion à distance sur cette

machine. Une étape de balyage des adresses sur Internet, à la recherche des machine proposant le service de connexion à distance, à donc été nécessaire avant toute attaque par dictionnaire.

Nous venons de mettre en évidence un processus d'attaque constitué de trois étapes : 1) balayage des adresses d'Internet à la recherche de machines proposant un service de connexion à distance, 2) attaque par dictionnaire à la recherche de couples valides et 3) intrusion. De plus, les attaquants ont organisé l'espace d'adressage sur Internet afin de spécialiser l'utilisation de chaque adresse à un type d'attaque. Cette cohérence est préservée dans le temps. Ils transfèrent donc les connaissances acquises lors de chacune des attaques par le biais de communications cachées, dans une base de connaissance qu'ils partagent. La suite de ce chapitre étudie plus en détail les étapes d'attaque par dictionnaire est d'intrusion.

4.5 Etude des attaques par dictionnaire

Cette section va nous permettre d'apporter des éléments de réponse aux questions suivantes : *les attaquants utilisent-ils tous des dictionnaires différents ? travaillent-ils en équipe pour découvrir des couples valides ? reviennent-ils après un échec ?*

Il est aisé de trouver, sur Internet, un outil d'attaque par dictionnaire et un dictionnaire de couples (nom d'utilisateur, mot de passe). Toutefois, fort de plusieurs années d'expériences, des attaquants peuvent être amenés à rédiger leur propre dictionnaire de couples. Pour réaliser cette rédaction, ils peuvent même s'appuyer sur les dictionnaires existants. Une fois établi, il y a de fortes chances pour que ce dictionnaire soit partagé par les membres de la communauté qui en est l'auteur. Pouvoir obtenir des informations sur les dictionnaires utilisés par les attaquants peut enrichir nos connaissances sur leur comportement et sur leurs habitudes. Cette connaissance peut nous permettre de réagir plus vite face à ce type d'attaques. Nous devons donc analyser les attaques par dictionnaire observées. Pour ce faire, nous introduisons la notion de *vocabulaire*. Le vocabulaire d'une attaque par dictionnaire observée est l'ensemble des couples (nom d'utilisateur, mot de passe) tentés. Quant à ces couples tentés, ils peuvent être assimilés aux *mots* du vocabulaire.

Dans cette section, nous exposons dans un premier temps un aperçu des attaques par dictionnaire observées en détaillant la notion de vocabulaire. Ensuite, nous présentons une définition d'une distance entre deux vocabulaires. Puis, à partir de la définition précédente de distance, nous classons les attaques par dictionnaire en confrontant deux à deux les vocabulaires. Pour finir, nous présentons les caractéristiques des attaques par dictionnaire observées.

4.5.1 Présentation des attaques par dictionnaire observées

L'attaquant s'appuie sur un dictionnaire pour mener une attaque par dictionnaire. Ce dictionnaire peut être plus ou moins volumineux. Par manque de temps, ou baisse de motivation, l'attaquant peut interrompre l'attaque avant son terme. Dans ce cas, la liste des couples tentés ne correspond pas exactement au contenu du dictionnaire de l'attaquant. N'ayant aucun moyen de savoir si l'attaquant a mené son attaque par dictionnaire à son terme, nous supposerons que l'ensemble des couples tentés

n'est qu'une partie du dictionnaire de l'attaquant. Deux attaques par dictionnaire peuvent employer le même dictionnaire. Si ces deux attaques ont été interrompues à des moments différents de leur progression, les deux ensembles de couples tentés seront ressemblants mais différents. Nous nommons vocabulaire associé à une attaque par dictionnaire l'ensemble des couples (nom d'utilisateur, mot de passe) tentés lors de cette attaque. Formellement, un vocabulaire $\mathcal{V}(e)$ associé à une attaque par dictionnaire $e \in Dico(\mathcal{T}, s)$ se définit comme suit :

$$\mathcal{V}(e) = \{couple(t)/t \in e\} \tag{4.12}$$

Deux vocabulaires dont l'intersection est vide ont de fortes chances d'avoir été générés depuis des dictionnaires différents. De même, deux vocabulaires dont l'intersection est importante ont de fortes chances d'avoir été générés depuis le même dictionnaire. Pouvoir identifier les similitudes entre vocabulaires nous permet de mieux parer aux attaques par dictionnaire, en les bloquant avant leur issue.

Caractéristique	Valeur
minimum	9
quantile 25%	25
médiane	157
quantile 75%	363
maximum	32078
moyenne	396

Fig. 4.13 – Caractéristiques statistiques des tailles des vocabulaires

Les vocabulaires sont plus ou moins importants. Par exemple, nous avons observé un vocabulaire d'une taille non significative (9 mots) et un autre d'une taille significative (32 078 mots). A titre d'illustration, la figure 4.13 présente les caractéristiques statistiques des tailles des vocabulaires. Une remarque importante concerne la moyenne et le troisième quartile (75%) : ils sont du même ordre de grandeur. Cela implique que les attaques par dictionnaire observées ont tendance à tester moins de 396 couples. De plus, le second quartile (25%) est relativement faible (25). La majorité des vocabulaires ont une taille comprise dans un intervalle réduit. La plupart des vocabulaires ne contiennent pas de répétitions. Cette caractéristique est compréhensible : si une identification échoue à un instant, il est peu probable qu'elle réussisse l'instant suivant. Pourtant, nous avons constaté que certains vocabulaires (peu nombreux) contiennent des couples en doublon. Pour quantifier le taux de répétition, nous avons calculé le rapport $r(e)$ entre le nombre de couples répétés au moins une fois et le nombre de couples différents, pour un vocabulaire e. Ce rapport est défini comme suit :

$$r(e) = \frac{|\{couple(t)/t \in e \wedge \exists t', t' \neq t \wedge couple(t) = couple(t')\}|}{|\{couple(t)/t \in e\}|} \tag{4.13}$$

Les valeurs de ce rapport pour les vocabulaires sont présentées sur la figure 4.14. Pour chaque vocabulaire reporté en abscisse, la valeur du rapport de ce vocabulaire est reporté en ordonnée. Les vocabulaires sont ordonnés par la valeur croissante de

leur rapport. Bien que la majorité des vocabulaires ne contiennent aucune répétition, nous en remarquons certains contenant uniquement des répétitions. La seule raison, à notre sens, qui permet de l'expliquer est un défaut des algorithmes qui ont permis à l'attaquant de créer son dictionnaire.

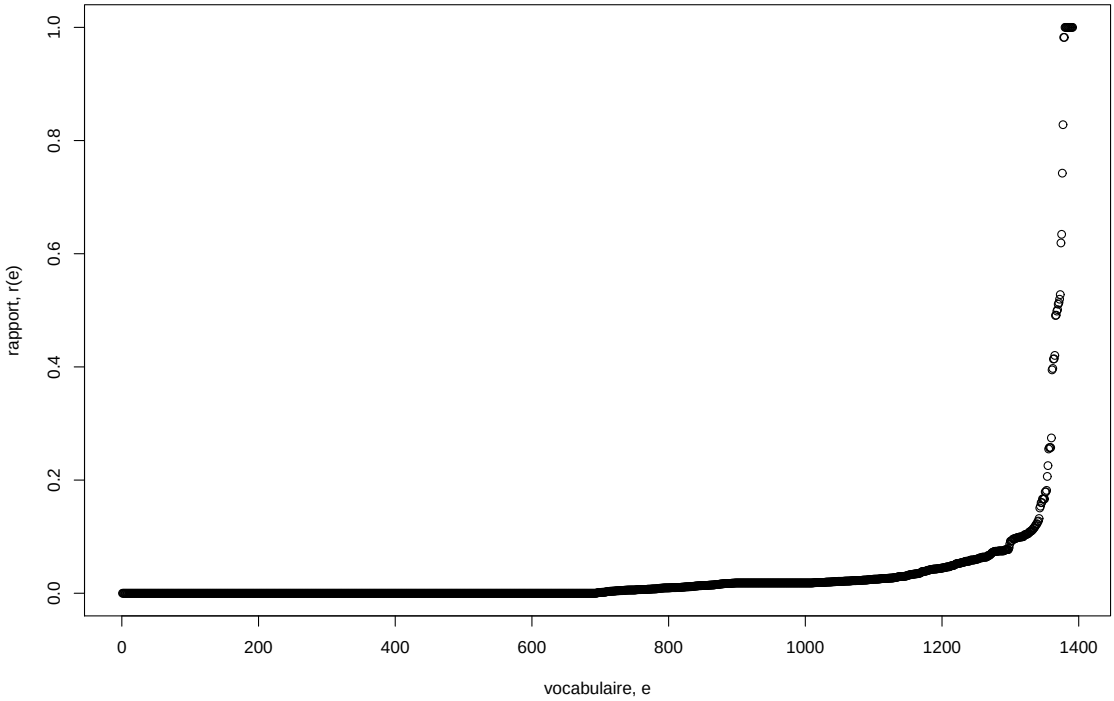


FIG. 4.14 – Rapport du nombre de couples différents répétés sur le nombre de couple différents pour chaque vocabulaire

Certains vocabulaires partagent des mots en commun et d'autres n'en ont aucun en commun. Ceci entraîne que l'intersection de tous les vocabulaires est vide. La figure 4.15 illustre, par un exemple, l'espace des mots ainsi que les parties couvertes par différents vocabulaires. Dans cet exemple, les vocabulaires $\mathcal{A}, \mathcal{B}, \mathcal{C}$ et $\mathcal{D}$ contiennent respectivement 6, 10, 2 et 8 mots. Le vocabulaire $\mathcal{C}$ est isolé : il ne partage aucun mot avec les autres vocabulaires. Les vocabulaires $\mathcal{A}$ et $\mathcal{B}$ partagent 1 mot en commun. Les vocabulaires $\mathcal{B}$ et $\mathcal{D}$ partagent 6 mots en commun. Plusieurs autres mots ne sont pas couverts par un vocabulaire. Pour cet exemple, il est vraisemblable que les vocabulaires $\mathcal{A}$ et $\mathcal{C}$ soient issus de deux dictionnaires sources différents du fait que leur intersection soit vide. En revanche, il est plausible d'affirmer que les vocabulaires $\mathcal{B}$ et $\mathcal{D}$ sont, quant à eux, issus du même dictionnaire source. Bien que $\mathcal{D}$ soit de taille inférieure à $\mathcal{B}$, il partage tout de même 75% de ses mots avec $\mathcal{B}$.

L'étude qualitative menée sur l'exemple 4.15 permet d'identifier des ressemblances ou des dissemblances entre les vocabulaires en les analysant deux à deux. Néanmoins, elle n'est pas adaptée pour l'étude des 1391 vocabulaires. Nous avons besoin d'un

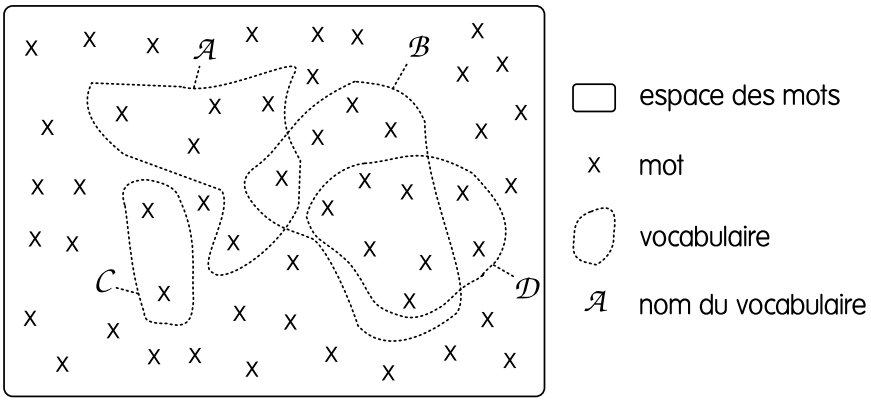


Fig. 4.15 – Exemple de couverture de vocabulaires

indicateur quantitatif pour étudier ce nombre important de vocabulaires. *Dictionnaire*, *vocabulaire*, *ressemblance*, *dissemblance*, ce champ lexical n'est pas sans rappeler l'analyse lexicographique. Dans ce cas, les vocabulaires des attaques par dictionnaire s'apparentent à des vocabulaires employés dans des textes littéraires. Nous pouvons ainsi profiter des travaux de ce domaine pour définir une distance entre deux vocabulaires issus d'attaques par dictionnaire. La section suivante présente des travaux sur la distance inter-textuelle.

4.5.2 Distance inter-textuelle

Une distance est un intervalle qui sépare deux éléments appartenant au même ensemble. Un ensemble muni d'une distance s'appelle un espace métrique. D'un point de vue spatial, une distance est un écart qui sépare deux objets. D'un point de vue temporel, une distance est un écart entre deux moments. La distance euclidienne est certainement la plus connue. Elle généralise l'application du théorème de Pythagore à un espace de dimension quelconque. Elle est utilisée dans le plan et l'espace. Une autre distance usuelle est la distance de la norme supérieure. Une dernière, quant à elle, peu usuelle, permet de quantifier la différence entre deux textes : la distance inter-textuelle. Penchons-nous plus en détail sur cette distance. Les travaux présentés dans [LL01] ont permis d'obtenir des résultats très intéressants dans le domaine des corpus. Dominique Labbé a appliqué cette distance, dans [Lab03], pour trancher sur le problème de la paternité de certaines œuvres que nous attribuons habituellement à Molière, au détriment de Corneille. Il a calculé la distance inter-textuelle entre toutes les œuvres de Molière et de Corneille. En étudiant ces distances inter-textuelles, il a montré que les œuvres de Corneille (*La Toison d'or*, *Andromède*, ...) sont nettement plus ressemblantes aux œuvres de Molière écrites par Corneille (*L'Avare*, *Dom Juan*, ...) que les œuvres de Molière écrites par lui-même (*Le Malade Imaginaire*, *Le Bourgeois Gentilhomme*, ...). Cet exemple anecdotique montre que ce genre de démarche peut aboutir à des résultats intéressants.

La distance employée dans [LL01] prend en compte des textes de tailles différentes. Soit deux textes, $\mathcal{A}$ et $\mathcal{B}$ tels que la taille du texte $\mathcal{B}$ est beaucoup plus importante que celle du texte $\mathcal{A}$. Sans prendre en compte cette différence, la distance obtenue

risque d'être trop importante. Cela ne pose pas de problème lorsqu'il est question de comparer un poème à un roman car une distance importante est justifiée par des genres de texte différents. Le problème est présent lorsqu'il faut comparer deux romans de tailles différentes. Afin de résoudre ce problème, dans [LL01], la taille du texte $\mathcal{B}$ est ramenée à celle du texte $\mathcal{A}$. La distance sera convenable si le texte $\mathcal{A}$ peut être considéré comme une version réduite du texte $\mathcal{B}$. Pour ce faire, les fréquences des mots du texte $\mathcal{B}$ sont pondérées par le rapport entre la taille du texte $\mathcal{A}$ et la taille du texte $\mathcal{B}$. Cette distance est définie comme suit :

$$D_{(a,b)} = \frac{\sum_{i \in j} |F(i,\mathcal{A}) - E(i,\mathcal{A},\mathcal{B})| + \sum_{i \in k} |F(i,\mathcal{A}) - E(i,\mathcal{A},\mathcal{B})| + \sum_{i \in l} |E(i,\mathcal{A},\mathcal{B})|}{D_j + D_k + D_l} \tag{4.14}$$

$$D_j = \sum_{i \in j} F(i,\mathcal{A}) \quad D_k = \sum_{i \in k} E(i,\mathcal{A},\mathcal{B}) \quad D_l = \sum_{i \in l} E(i,\mathcal{A},\mathcal{B})$$

Nous avons utilisé les notations suivantes :

- $F(i,\mathcal{A})$ est la fréquence du mot i dans le texte $\mathcal{A}$.
- $E(i,\mathcal{A},\mathcal{B})$ est la fréquence attendue du mot i dans le texte $\mathcal{A}$, par rapport au texte $\mathcal{B}$. Cette dernière est obtenue par la pondération de la fréquence de i dans $\mathcal{B}$, par le coefficient $U_{(a,b)} = \frac{|\mathcal{A}|}{|\mathcal{B}|}$.
- j est l'ensemble des mots de $\mathcal{A}$ dont la fréquence est égale ou supérieure à celle attendue ($F(i,\mathcal{A}) \geq E(i,\mathcal{A},\mathcal{B})$).
- k est l'ensemble des mots de $\mathcal{A}$ dont la fréquence est inférieure à celle attendue ($F(i,\mathcal{A}) < E(i,\mathcal{A},\mathcal{B})$).
- l est l'ensemble des mots absents de $\mathcal{A}$, dont la fréquence attendue est supérieure à 1 ($F(i,\mathcal{A}) = 0 \wedge E(i,\mathcal{A},\mathcal{B}) \geq 1$).

La figure 4.16 illustre les différents ensembles. La section suivante adapte les principes présentés dans cette section pour définir une distance entre deux vocabulaires.

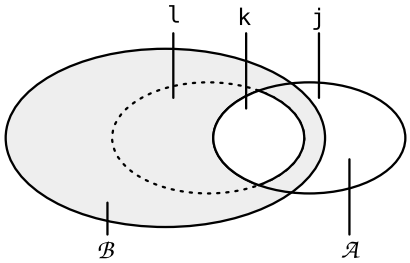


FIG. 4.16 – Représentation des ensembles utilisés dans le calcul de la distance inter-textuelle

4.5.3 Définition de la distance entre deux vocabulaires

Nous utilisons une distance semblable à celle définie dans [LL01] pour comparer des vocabulaires. Toutefois, la distance 4.14 de la section 4.5.2 ne peut pas être utilisée telle quelle pour comparer deux à deux les vocabulaires. La principale raison concerne

les répétitions. Dans un texte littéraire, ces répétitions sont inévitables. En reprenant les textes $\mathcal{A}$ et $\mathcal{B}$ précédents, les fréquences pondérées des mots du texte $\mathcal{B}$ sont, en majorité, supérieures à 1. Parmi les vocabulaires des attaquants, certains contiennent effectivement beaucoup de répétitions (cf. figure 4.14). Toutefois, le nombre de vocabulaires contenant beaucoup de répétitions est minime et les autres vocabulaires présentent peu – voire pas du tout – de répétitions. Aussi, les fréquences des mots des vocabulaires sont souvent égales à 1. Donc, si on utilise la méthode précédente pour comparer deux vocabulaires de taille très différente, les fréquences pondérées des mots du vocabulaire le plus gros seront nettement inférieures à 1 et donc non comparables aux fréquences de ces mêmes mots dans le vocabulaire plus petit. Ceci entraînerait une distance trop importante, quelle que soit la ressemblance véritable entre ces deux vocabulaires. Sans pour autant ignorer les répétitions, nous n'allons pas leur attacher trop d'importance. La distance que nous proposons d'utiliser est définie comme suit :

$$D_{(\mathcal{A},\mathcal{B})} = \frac{\sum_{m\in\mathcal{M}} (F(m,\mathcal{A}) - F(m,\mathcal{B})) \cdot \sum_{m\in\mathcal{N}} (F(m,\mathcal{B}) - F(m,\mathcal{A}))}{\sum_{m\in\mathcal{A}} F(m,\mathcal{A}) \cdot \sum_{m\in\mathcal{B}} F(m,\mathcal{B})} \tag{4.15}$$

$$\mathcal{M} = \{m \in \mathcal{A}/F(m,\mathcal{A}) \geq F(m,\mathcal{B})\} \quad \mathcal{N} = \{m \in \mathcal{B}/F(m,\mathcal{B}) > F(m,\mathcal{A})\}$$

Cette distance vérifie les propriétés suivantes. Le numérateur est supérieur ou égal à 0 et inférieur ou égal au dénominateur. Cette distance est donc comprise dans l'intervalle des réels $[0;1]$. Si l'un des deux vocabulaires est inclus dans l'autre, elle vaut 0. Si l'intersection des deux vocabulaires est vide, elle vaut 1. La question qui nous intéresse est "que vaut cette distance si $\alpha\%$ des mots d'un des deux vocabulaires est partagé avec l'autre vocabulaire ?". Par le raisonnement suivant, nous allons y répondre.

$$(1-s) = \frac{\sum_{m\in\mathcal{M}} (F(m,\mathcal{A}) - F(m,\mathcal{B}))}{\sum_{m\in\mathcal{A}} F(m,\mathcal{A})} \tag{4.16}$$

$$(1-s') = \frac{\sum_{m\in\mathcal{N}} (F(m,\mathcal{B}) - F(m,\mathcal{A}))}{\sum_{m\in\mathcal{B}} F(m,\mathcal{B})} \tag{4.17}$$

Soit s (respectivement s') la proportion des mots de $\mathcal{A}$ (respectivement $\mathcal{B}$) partagés avec $\mathcal{B}$ (respectivement $\mathcal{A}$). Notons $s_{max} = \max(s, s')$ et $s_{min} = \min(s, s')$. La quantité s_{min} est la proportion des mots du plus gros vocabulaire partagés avec le plus petit. Lorsque s_{min} tend vers 1 alors le gros vocabulaire tend à être identique au petit vocabulaire. s_{min} mesure la ressemblance entre les deux vocabulaires. De la même manière, la quantité s_{max} est la proportion des mots du plus petit vocabulaire partagés avec le plus gros. Lorsque s_{max} tend vers 1 alors le petit vocabulaire tend à être couvert par le gros vocabulaire. s_{max} mesure la couverture du plus petit vocabulaire par le plus gros. L'évolution du produit de s_{min} par s_{max} est présenté à la figure 4.17.

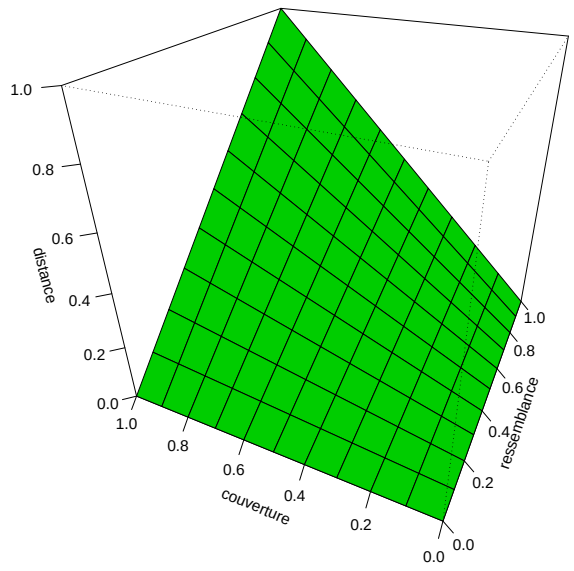


FIG. 4.17 – Evolution de la distance en fonction de la couverture et de la ressemblance

Deux attaques par dictionnaire réalisées à partir du même dictionnaire source peuvent être interrompues par les attaquants à des moments différents. Nous observons sur le pot de miel deux vocabulaires de taille différente. Ce n'est donc pas la mesure de ressemblance qui va nous permettre de déterminer si ces deux vocabulaires sont issus du même dictionnaire source mais plutôt la mesure de couverture. Nous allons définir un critère de regroupement en fonction de la couverture du plus petit vocabulaire par le plus gros.

La distance s'exprime ainsi : $D_{(\mathcal{A},\mathcal{B})} = (1-s)(1-s') \geq (1-s_{max})^2$, car $s_{max} \geq s_{min}$. Si $\alpha \geq D_{(\mathcal{A},\mathcal{B})}$, il s'ensuit $\alpha \geq (1 - s_{max})^2$. Donc, toute distance $D_{(\mathcal{A},\mathcal{B})}$ inférieure à α implique que $(1 - \sqrt{\alpha}) \cdot 100\%$ des mots du plus petit vocabulaire sont recouverts par le plus gros. Cette distance va être calculée sur les vocabulaires, deux à deux, afin de les regrouper en grappes, en fonction du seuil α. Ainsi, pour tous les couples de vocabulaires d'une grappe, le plus petit des deux partagera au moins $(1 - \sqrt{\alpha}) \cdot 100\%$ de ces mots avec le plus gros.

Pour réaliser ce traitement, les distances entre les vocabulaires, pris deux à deux, sont reportées dans une matrice carrée. Cette matrice possède autant de lignes et de colonnes que de vocabulaires. Nous affecterons, à la cellule (i, j) de cette matrice, la distance entre les vocabulaires i et j. Cette matrice possède la particularité d'être symétrique. Elle est induite de la relation suivante : $D_{(\mathcal{A},\mathcal{B})} = D_{(\mathcal{B},\mathcal{A})}$. Cette matrice nous sert de point de départ pour réaliser un regroupement en grappe (en anglais, *clustering*). Ce regroupement a été réalisé avec le logiciel `R`[R-D07], et, plus particulièrement, la fonction `hclust`. Son principe est abordé dans la section suivante, avant d'être appliqué aux données.

4.5.4 Partitionnement des données

Le partitionnement des données (en anglais, *clustering*) est une méthode de classification non supervisée. Elle permet de classer des données en les regroupant en grappes, de telle manière que les données au sein d'une même grappe présentent beaucoup de similarité. Cette méthode peut être utilisée pour préparer des données à un autre algorithme ou pour réduire la taille des données. Elle est largement utilisée en biologie pour l'étude des séquences de protéines, par exemple.

Les informations de départ sont l'ensemble E des éléments à classer, une fonction $d(e_i, e_j)$ et un seuil s. Cette fonction retourne la distance entre deux éléments de E. La méthode du partitionnement des données permet de regrouper les éléments de E en grappes, de telle manière que la distance entre les éléments d'une même grappe soit inférieure au seuil donné.

A titre d'illustration, utilisons la figure 4.18. L'ensemble de départ E, représenté au sommet de la figure, est composé de carrés, ronds et d'étoiles. Chacune de ces formes possède une taille qui peut être petite, moyenne ou grande. Comme fonction retournant la distance entre deux formes géométriques, nous utiliserons :

$$d(e_i, e_j) = \begin{cases} 0, & \text{si } e_i \text{ et } e_j \text{ ont la même forme et la même taille ;} \\ 1, & \text{si } e_i \text{ et } e_j \text{ ont la même forme mais des tailles différentes ;} \\ 2, & \text{si } e_i \text{ et } e_j \text{ n'ont ni la même forme ni la même taille.} \end{cases} \tag{4.18}$$

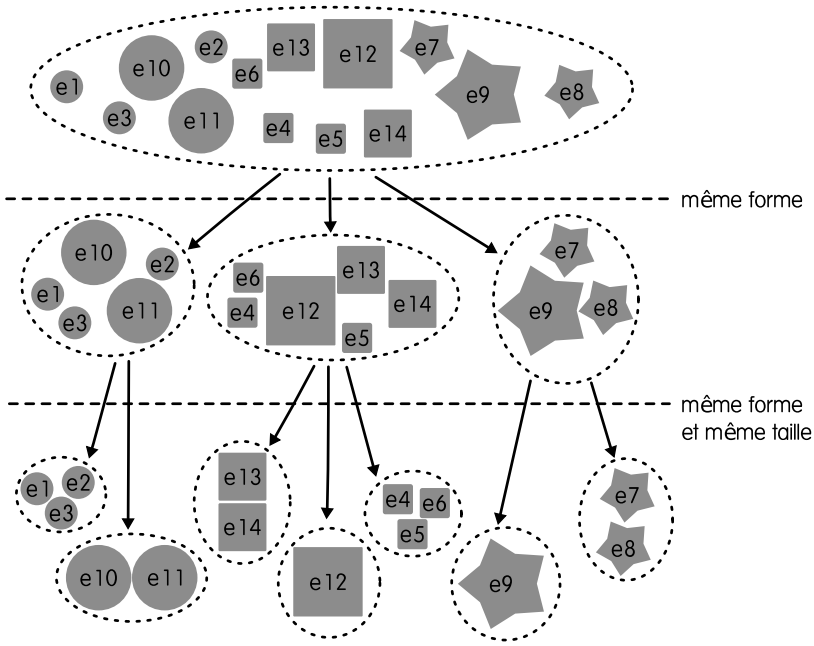


Fig. 4.18 – Exemple de partitionnement

Si notre seuil est "les éléments d'une même grappe doivent avoir la même forme", autrement dit $d(e_i, e_j) \leq 1$, alors nous regroupons ensemble les carrés, les ronds et les étoiles. Par contre, si nous utilisons un seuil beaucoup plus restrictif, $d(e_i, e_j) = 0$,

alors nous éclaterons les grappes précédemment trouvées pour ne regrouper ensemble que les formes strictement égales.

La fonction de distance doit être bien définie pour que les regroupements aient un sens. Dans l'exemple précédent, nous aurions pu utiliser une fonction qui regroupe d'abord par taille. Les résultats obtenus auraient été différents des précédents. Un soin particulier doit donc être apporté à la définition de cette fonction.

A partir de cette fonction de distance, plusieurs algorithmes peuvent être utilisés pour effectuer les regroupements. Certains algorithmes, du type *bottom-up*, fonctionnent par agglomération : les grappes sont successivement unies pour obtenir des grappes plus grandes. Un autre type d'algorithmes, *top down*, utilise une approche descendante : les grappes sont successivement divisées pour obtenir de plus petites grappes, en considérant, pour le départ, une seule grappe constituée de tous les éléments. Etant donné la quantité de données à traiter dans notre cas, nous utiliserons principalement une approche descendante, afin d'assurer un minimum de regroupements.

Il existe deux principales représentations des grappes. La première est nommée dendrogramme. Un dendrogramme est un diagramme arborescent qui montre, pour chaque élément, l'ensemble des éléments qui partagent les mêmes ancêtres. La figure 4.18 en est un exemple. A cette représentation, nous préferons la représentation matricielle. Une telle représentation utilise une matrice pour laquelle les lignes sont indicées par les éléments et les colonnes sont indicées par les mêmes éléments. Une cellule de la matrice contient la distance entre les éléments de la ligne et de la colonne correspondantes. Il existe plusieurs permutations des colonnes et des lignes de cette matrice. Afin d'en améliorer la lisibilité, une permutation choisie est une permutation qui va mettre côte à côte les éléments qui sont proches. Enfin, pour rendre une telle représentation plus visuelle, le contenu des cellules est représenté par une couleur. Plus intense est cette couleur, plus petite est la distance correspondante. La matrice correspondant à la figure 4.18 est donnée en figure 4.19.

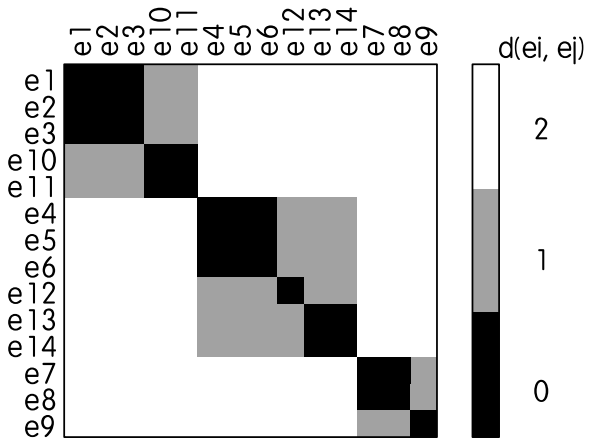


Fig. 4.19 – Exemple de représentation matricielle

4.5.5 Classement des vocabulaires

Nous voulons regrouper ensemble des vocabulaires très similaires. Nous supposerons que deux vocabulaires sont similaires si 75% des mots du plus petit sont partagés avec le plus gros. Ainsi, le critère s'exprime de la manière suivante : $\alpha = (1-0,75)^2 = 0,0625 \geq D_{(\mathcal{A},\mathcal{B})}$.

En appliquant ce critère, nous avons ainsi identifié, pour l'ensemble des 1391 vocabulaires nommé $\mathcal{G}_d(\mathcal{T}, s)$, 434 grappes. Cet ensemble est une partition de l'ensemble des attaques par dictionnaire observées, $\mathcal{G}_d(\mathcal{T}, s) \in \mathcal{P}(Dico(\mathcal{T}, s))$. La figure 4.20 est une présentation matricielle du partitionnement des vocabulaires en grappes (cf. section 4.5.4). Rappelons que, sur cette représentation, une colonne représente un vocabulaire, de même pour les lignes. L'intersection d'une colonne, représentant le vocabulaire $\mathcal{A}$, et d'une ligne, représentant le vocabulaire $\mathcal{B}$ contient un point en nuance de gris. La nuance de gris employée dépend de la distance entre $\mathcal{A}$ et $\mathcal{B}$: plus cette distance est proche de 0, plus le point est sombre. Les colonnes et lignes ont été ordonnées de manière à placer côte à côte les vocabulaires proches, donc similaires.

Cette image met en évidence 2 grappes imposantes, symbolisées par deux grandes tâches sombres. A elles seules, ces grappes contiennent plus du quart des vocabulaires. La plus grande grappe contient 187 vocabulaires. 183 d'entre eux sont strictement identiques et composés de 9 mots et les 4 autres sont eux aussi strictement identiques et composés de 363 mots. Toutes les distances, dans cette grappe valent 0 : les vocabulaires de 9 mots sont inclus dans ceux de 363 mots. Pour la grappe suivante, elle est composée de 161 vocabulaires. 111 d'entre eux ont une taille de 168 mots et sont identiques. Les autres vocabulaires de cette grappe ont des tailles variables, entre 9 et 778 mots.

La grappe suivante contient 23 vocabulaires. Tous ces vocabulaires ont des tailles différentes variant entre 20 mots et 228 mots, pour une moyenne de 104 mots par vocabulaire. Cette grappe reflète l'intérêt du regroupement en grappes. Avec une analyse vocabulaire par vocabulaire, nous n'aurions pas forcément uni ces vocabulaires.

Globalement, les grappes correspondent à des attaques par dictionnaire qui utilisent un même dictionnaire source. Une grappe peut donc identifier une communauté d'attaquants. Pour les grappes imposantes, soit la communauté est importante, soit cette communauté est très active. Quant aux petites grappes, elles correspondent à des attaquants isolés. Par exemple, les deux grappes importantes présentées dans la figure 4.20 sont proches l'une de l'autre : les contours ne sont pas foncés mais d'une nuance assez sombre. Elles identifient chacune une communauté qui utilise le même dictionnaire d'origine.

Le dictionnaire source a pu évoluer, par ajout ou suppression de couples. La partie stable est nommée le *cœur de dictionnaire*. Peut-être ces deux communautés sont les mêmes et le cœur de dictionnaire qu'elles utilisent a évolué dans le temps ? En utilisant un seuil plus souple, nous aurions rassemblé ces vocabulaires. Le revers d'un seuil plus bas est que nous aurions, par la même occasion, rassemblé des vocabulaires issus de communautés différentes.

Rappelons que l'objectif de cette section est d'analyser les dictionnaires employés par les attaquants. A présent, nous disposons uniquement, pour chaque attaque par dictionnaire, des connexions observées sur le pot de miel. Ces connexions ne reflètent qu'une partie plus ou moins grande du dictionnaire employé par l'attaquant. De plus,

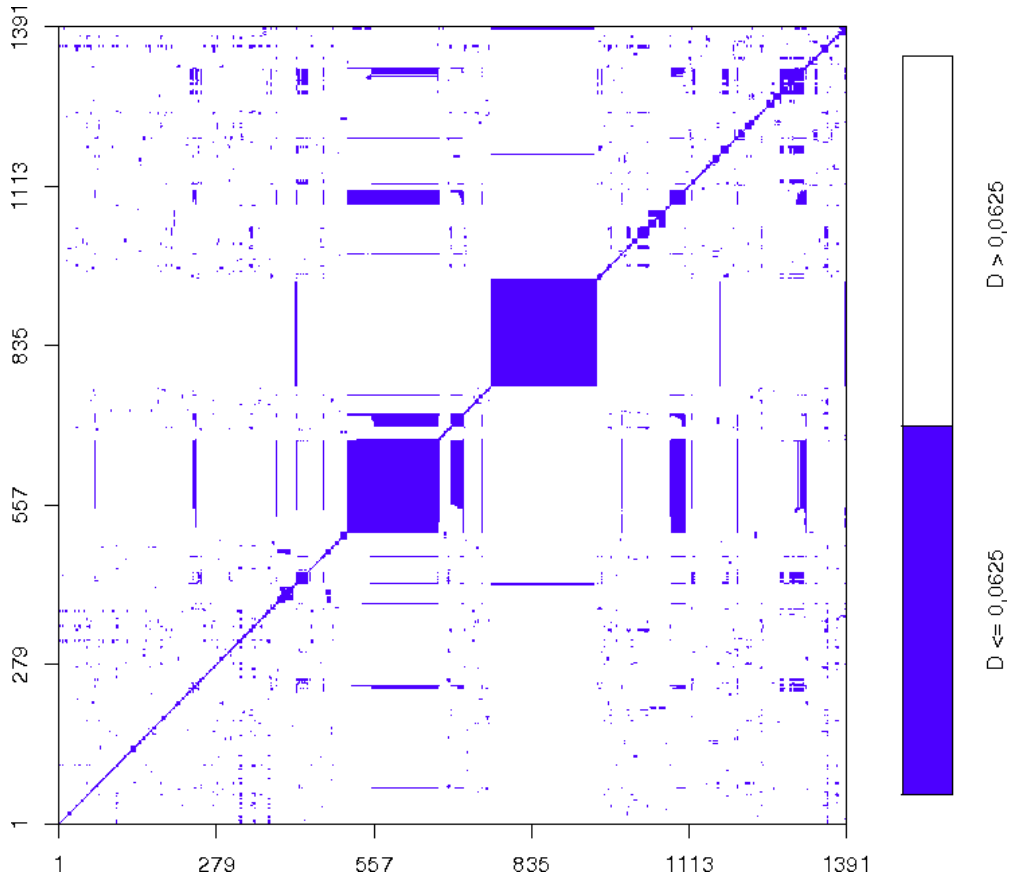


FIG. 4.20 – Distances entre les vocabulaires

un attaquant peut modifier à sa guise un dictionnaire pour l'enrichir des couples (nom d'utilisateur, mot de passe) qui semblent pertinents. Le regroupement des vocabulaires en grappes nous montre que beaucoup d'attaquants partagent le même dictionnaire. La section suivante tente d'obtenir des caractéristiques sur le dictionnaire associé à chaque grappe.

4.5.6 Identification des cœurs de dictionnaire

Nous disposons à présent des grappes de vocabulaires. Au sein de chaque grappe, les vocabulaires sont similaires. Ils peuvent avoir été générés à partir du même dictionnaire ou à partir de dictionnaires proches : l'attaquant peut avoir modifié un dictionnaire en l'enrichissant de nouveaux couples ou encore il peut avoir appauvri ce dictionnaire en ciblant uniquement certains couples. Nous cherchons à caractériser le dictionnaire source à l'origine de ces dictionnaires modifiés. Bien entendu, nous ne pouvons pas reconstituer ce dictionnaire source dans son intégralité car certains de ses couples peuvent ne pas avoir été tentés sur le pot de miel. Nous allons donc essayer de reconstituer une partie de ce dictionnaire que nous appelons *cœur de dictionnaire*.

Nous ne pouvons affirmer avec exactitude qu'un couple qui a été tenté uniquement lors d'une seule attaque de la grappe appartient au cœur de dictionnaire. Le cœur de dictionnaire associé à une grappe ne peut donc pas être considéré comme l'union

des vocabulaires de la grappe. De la même manière, il serait trop restrictif d'écarter les couples tentés par plusieurs attaques de la grappe mais pas toutes. Le cœur de dictionnaire associé à une grappe ne peut donc pas être considéré comme l'intersection des vocabulaires de la grappe.

Nous considèrons que le cœur de dictionnaire, $\mathcal{D}_d(g)$ associé à une grappe g est l'ensemble des mots partagés par au moins deux vocabulaires de cette grappe. Il est défini de la manière suivante :

$$\mathcal{D}_d(g) = \{m/\exists(e, e') \in g^2, e \neq e'/m \in e \wedge m \in e'\} \tag{4.19}$$

L'exemple de la figure 4.21 présente quatre vocabulaires A, B, C et D. Le cœur de dictionnaire associé est identifié et présenté avec une surface grise. Il illustre la formule précédente.

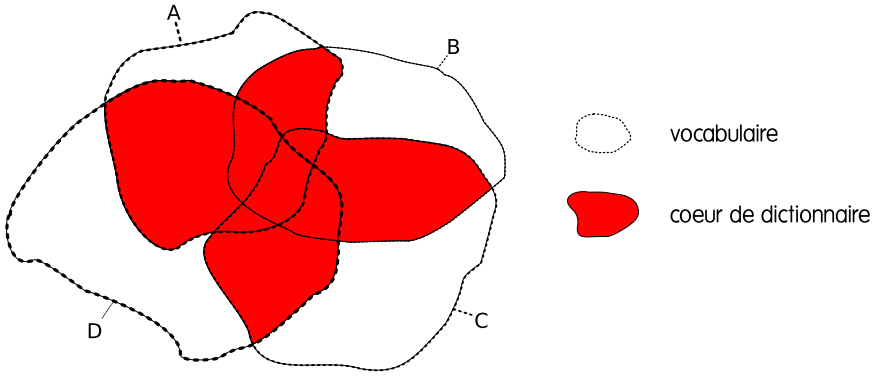


FIG. 4.21 – Exemple de construction du cœur de dictionnaire pour une grappe

A partir des cœurs de dictionnaire des 434 grappes, nous avons déterminé la nature des couples employés. Un couple est composé d'un nom d'utilisateur et d'un mot de passe. Le nom d'utilisateur et le mot de passe peuvent être des mots d'une langue parlée telle que le français ou l'anglais. Ils peuvent aussi être des mots sans aucun sens, issus d'une génération combinatoire de lettres. Pour pouvoir déterminer la nature des couples employés, nous avons récupéré un dictionnaire des langues française et anglaise notée $\mathcal{L}$, disponibles avec le logiciel `ISPELL`[KWBS]. Nous avons alors, pour chaque cœur de dictionnaire, confronté les noms d'utilisateur et mots de passe tentés aux mots des dictionnaires français et anglais, afin d'établir des profils.

Pour établir ces profils, nous nous sommes intéressés à quatre proportions. Tout d'abord, il est important de prêter attention à la proportion des répétitions des noms d'utilisateur dans un cœur de dictionnaire C, notée $P_{rep}(C)$. Ensuite, sur la base du dictionnaire des langues française et anglaise, nous avons déterminé la proportion des noms d'utilisateur (respectivement mots de passe) correspondant à des noms communs ou des noms propres, notée $P_{nom}(C)$ (respectivement $P_{passe}(C)$). Pour finir, nous avons défini la proportion des couples pour lesquels le nom d'utilisateur est identique au mot de passe, notée $P_{id}(C)$. Ces quatre proportions sont définies comme suit :

$$P_{rep}(C) = \frac{|\{c \in C / \exists c' \in C, c \neq c' \wedge nom(c) = nom(c')\}|}{|C|} \tag{4.20}$$

$$P_{nom}(C) = \frac{|\{c \in C / nom(c) \in \mathcal{L}\}|}{|C|} \tag{4.21}$$

$$P_{passe}(C) = \frac{|\{c \in C / passe(c) \in \mathcal{L}\}|}{|C|} \tag{4.22}$$

$$P_{id}(C) = \frac{|\{c \in C / passe(c) = nom(c)\}|}{|C|} \tag{4.23}$$

Les résultats obtenus sont reportés dans le tableau 4.22. Les quatres dernières colonnes de ce tableau représentent les proportions précédentes. La présence d'une coche dans une cellule signale que la proportion correspondante est supérieure à 25%. Chaque ligne correspond alors à un profil de cœur de dictionnaire. Pour les 4 proportions considérées, 16 profils différents peuvent exister. Quant à la première colonne, $|\{C\}|$, elle correspond au nombre de cœurs de dictionnaire ayant le profil correspondant.

$\|\{C\}\|$	$P_{rep}(C) \geq 25\%$	$P_{nom}(C) \geq 25\%$	$P_{id}(C) \geq 25\%$	$P_{passe}(C) \geq 25\%$
181	✓	✓	✓	✓
76				
74	✓		✓	
30	✓	✓		
23	✓			
20				✓
14	✓	✓		✓
9	✓		✓	✓
5	✓	✓	✓	
2	✓			✓

Fig. 4.22 – Profils des cœurs de dictionnaire

Tout d'abord, seulement 10 des 16 profils sont représentés. Le premier profil est prédominant et le dernier est négligeable. Nous constatons que les cœurs de dictionnaire contiennent des couples qui ne sont pas totalement générés de manière combinatoire. Ils ont tous été générés à partir d'une logique précise. La plupart des noms d'utilisateur et des mots de passe font partie des langues française et anglaise. De plus, un bon nombre de cœurs de dictionnaire utilise des couples composés de nom d'utilisateur et mot de passe identiques. Certains noms d'utilisateur sont testés plusieurs fois. L'obstination des attaquants à tester plusieurs fois le même nom d'utilisateur indique qu'ils sont intéressés par les noms d'utilisateur fréquemment rencontrés dans les systèmes informatiques.

Nous pouvons supposer que les attaquants n'épuisent pas leurs efforts dans des attaques qui sont connues pour donner peu de résultats, voire aucun. Si les attaques par dictionnaire sont encore largement utilisées, alors cela signifie qu'elles portent leurs fruits. Les attaquants qui les emploient font sûrement bénéfice du comportement de

beaucoup d'utilisateurs : employer des mots de passe simples. Donc, tant que les mots de passe ne seront pas tous difficiles à deviner, les attaquants estimeront qu'il est utile de passer du temps à employer les attaques par dictionnaire pour les deviner. De plus, cette constatation met en évidence l'utilité de la déclaration de couples (nom d'utilisateur, mot de passe) simples à deviner. Nous pouvons supposer que ce n'est pas ce point qui va mettre la puce à l'oreille de l'attaquant pour soupçonner la présence du pot de miel.

4.6 Etude des intrusions

Une attaque par dictionnaire n'est pas une fin en soi. Ce n'est qu'une étape du processus d'intrusion, comme le suggère le temps entre la découverte du couple valide et la première intrusion (cf. section 4.1), qui est élevé. L'objectif de l'attaquant est plus précisément d'obtenir un moyen d'accès sur sa cible. Après la découverte d'un couple (nom d'utilisateur, mot de passe) valide, par une attaque par dictionnaire, nous observons une connexion réussie avec saisie de commandes. Cette connexion s'apparente à une intrusion. Elle correspond à la deuxième étape du processus d'intrusion. Parmi les 1 940 sessions identifiées précédemment, 210 sont considérées comme telles. Ces intrusions font l'objet de la présente section.

Le nombre d'adresses différentes à l'origine de ces 210 intrusions s'élève à 57. Elles ont ciblé 21 comptes différents. Le tableau 4.23 présente, pour chaque compte, les nombres d'intrusions, d'adresses et de mots de passe associés. Une première remarque concerne les mots de passe. Plusieurs mots de passe peuvent être associés à un même compte. Effectivement, nous avons constaté que l'une des premières actions réalisées par l'intrus est de changer le mot de passe associé au compte qu'il vient d'usurper. Une remarque importante porte sur les adresses ayant réalisé ces intrusions. Aucune de ces adresses n'a été utilisée pour mener une attaque par dictionnaire. Or, étant donné le nombre important d'intrusions disponibles, nous pouvons fortement supposer que les attaquants sur Internet spécialisent les machines à leur disposition pour certaines attaques. En poussant un peu plus le raisonnement, nous pourrions supposer que les attaquants se regroupent en communautés pour mener leurs actions de manière concertée en utilisant différentes machines pour mener leurs attaques.

Dans les paragraphes suivants, nous analysons le comportement des intrus. Tout d'abord, nous allons identifier les différents groupes d'intrus à l'origine des 210 intrusions. Ces groupes sont les *communautés d'attaquants*. Ensuite, nous caractérisons la nature des intrus. Plus précisément, nous déterminons s'ils sont des êtres humains ou des outils automatiques. Puis, nous présentons les différentes activités qu'ils ont réalisées au sein du pot de miel. Pour finir, nous mettons en évidence leur niveau de compétence.

4.6.1 Identification des différentes communautés

Deux intrusions réalisées par le même intrus dans un même laps de temps ont de fortes chances d'être corrélées. Un couple valide est trouvé lors d'une attaque par dictionnaire. L'intrus utilise cette information pour se connecter directement au pot de miel. Il peut aussi partager cette information avec un complice. Etudier une intrusion

Compte	Nombre d'intrusions	Nombre de mots de passe	Nombre d'adresses
C_1	11	2	6
C_2	3	2	2
C_3	39	2	10
C_4	2	2	1
C_5	3	2	2
C_6	22	2	7
C_7	21	2	2
C_8	32	2	3
C_9	14	2	6
C_{10}	2	1	2
C_{11}	3	2	2
C_{12}	17	3	3
C_{13}	4	2	1
C_{14}	3	2	2
C_{15}	1	1	1
C_{16}	3	2	1
C_{17}	19	1	7
C_{18}	2	2	1
C_{19}	1	1	1
C_{20}	5	1	3
C_{21}	2	1	1

FIG. 4.23 – Nombre d'intrusions par compte

indépendamment des autres permet de comprendre pourquoi elle a été réalisée. Toutefois, situer cette intrusion au sein d'une communauté permet de mieux caractériser le comportement de groupe des attaquants de cette communauté.

Pour mener à bien l'identification des communautés, nous nous basons sur les intrusions identifiées à la section 4.3. Une intrusion est un ensemble de connexions réalisées par une machine (celle de l'intrus) et à destination d'une même machine du pot de miel (la cible de l'intrus). Rappelons que chaque connexion est caractérisée par l'adresse de la machine à son origine, l'adresse de la machine destination (du pot de miel), le couple (nom d'utilisateur, mot de passe) tenté, un booléen indiquant si cette connexion a réussi ainsi qu'un booléen indiquant si cette connexion est suivie par une saisie de commande ou non.

Le pot de miel est caractérisé par son état. Au début de sa première intrusion, l'intrus possède une connaissance limitée sur l'état du pot de miel. Cette connaissance se résume à un ensemble de couples (nom d'utilisateur, mot de passe) valides obtenus suite à une attaque par dictionnaire. Pour mener à bien son activité, sitôt connecté, il doit enrichir ses connaissances. Au fil de cette activité, l'état du pot de miel peut être affecté. Les connaissances de l'intrus sur sa cible vont évoluer. Elles peuvent aussi être partagées avec ses complices. L'intrus et ses complices forment une communauté d'attaquants. A tel point que la prochaine intrusion peut être réalisée par un autre intrus, avec les connaissances acquises par le précédent.

La figure 4.24 illustre l'évolution des connaissances de l'intrus, au cours du temps.

Pour sa première intrusion 1, l'intrus d'adresse @1 utilise le couple C_1. Aussitôt connecté, il liste les fichiers du répertoire courant, `ls`, et change le mot de passe du compte courant C_1, `passwd`. A ce stade, l'état du pot de miel a changé : le mot de passe du couple C_1 a été modifié pour devenir le couple C_2. L'intrus, qui est à l'origine de cette activité, fait évoluer ses connaissances en même temps. Plus tard, un autre intrus, complice du premier, vient pour réaliser l'intrusion 2. Il se connecte depuis la machine d'adresse @2 avec le couple C_2. Il lance alors les commandes `ls` et `pwd`. Dans cet exemple, nous observons deux intrusions différentes réalisées par deux intrus appartenant à la même communauté. Pour identifier les différentes communautés, nous devons recomposer une partie des connaissances ayant permis chaque intrusion.

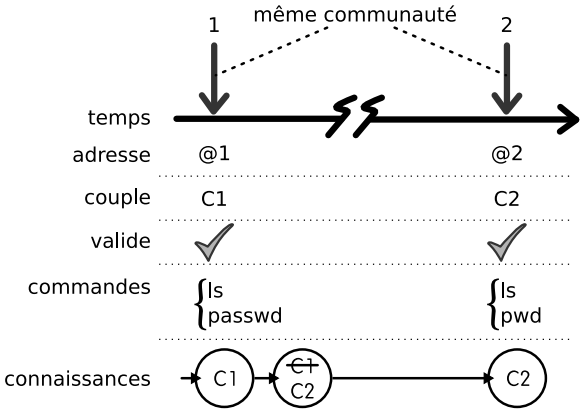


Fig. 4.24 – Evolution des connaissances de l'intrus

Nous sommes incapables de reconstituer l'intégralité des connaissances de l'intrus. En revanche, nous pouvons facilement identifier l'adresse de l'intrus, la liste des noms d'utilisateur et mots de passe valides qu'il a utilisées pour l'intrusion et les traces de l'ensemble des activités de l'intrus sur le pot de miel. La partie des connaissances de l'intrus qui nous intéresse est nommée *bagage*. Le bagage de l'intrusion i est noté $\mathcal{B}(i)$. Si deux intrusions possèdent des connaissances en commun, alors les bagages correspondants peuvent être très proches. Si deux intrusions ne possèdent aucune connaissance en commun, alors les bagages correspondants sont très éloignés. Inversement, nous supposons que deux bagages qui sont très proches identifient deux intrusions ayant des connaissances en commun et deux bagages qui sont éloignés peuvent identifier des intrusions n'ayant aucune connaissance en commun. L'écart entre les bagages nous permettra d'identifier les communautés d'attaquants : deux bagages proches identifient la même communauté.

La distance entre deux bagages doit refléter la quantité de connaissances partagée. Nous allons définir une distance entre deux bagages $\mathcal{B}_i$ et $\mathcal{B}_j$. Elle sera notée $D_b(\mathcal{B}_i, \mathcal{B}_j)$. Pour faciliter la notation, nous utilisons une distance générique, notée $\mathcal{D}_e(\mathcal{E}_i, \mathcal{E}_j)$, entre deux parties $\mathcal{E}_i$ et $\mathcal{E}_j$ d'un même espace. Notons $I_a(\mathcal{B})$ l'ensemble des adresses contenues dans le bagage $\mathcal{B}$, $I_n(\mathcal{B})$ l'ensemble des noms d'utilisateur contenus dans $\mathcal{B}$ et $I_m(\mathcal{B})$ l'ensemble des mots de passe contenus dans $\mathcal{B}$. La distance $D_b(\mathcal{B}_i, \mathcal{B}_j)$, est définie comme suit :

$$D_b(\mathcal{B}_i, \mathcal{B}_j) = \frac{\mathcal{D}_e(I_a(\mathcal{B}_i), I_a(\mathcal{B}_j)) + \mathcal{D}_e(I_n(\mathcal{B}_i), I_n(\mathcal{B}_j)) + \mathcal{D}_e(I_m(\mathcal{B}_i), I_m(\mathcal{B}_j))}{|I_a(\mathcal{B}_i)| + |I_a(\mathcal{B}_j)| + |I_n(\mathcal{B}_i)| + |I_n(\mathcal{B}_j)| + |I_m(\mathcal{B}_i)| + |I_m(\mathcal{B}_j)|} \quad (4.24)$$
$$\mathcal{D}_e(\mathcal{E}_i, \mathcal{E}_j) = |\mathcal{E}_i - \mathcal{E}_j| \cdot |\mathcal{E}_j - \mathcal{E}_i|$$

Nous disposons d'une distance pour mesurer l'écart entre intrusions, en terme de bagages. Elle a été appliquée sur l'ensemble des intrusions, prises deux à deux. Les valeurs obtenues ont été reportées dans une matrice. Cette matrice est carrée. Elle possède autant de lignes et de colonnes que d'intrusions, soit 210. A chaque ligne et colonne est associée une intrusion. Une représentation en est donnée à la figure 4.25. Dans cette représentation, un point en nuance de gris correspond à la distance entre les bagages de la ligne et de la colonne correspondantes. Plus ce point est sombre, plus petite est cette distance. Les colonnes et les lignes ont été permutées de manière à placer un bagage à côté de ceux qui lui sont proches. La première remarque concerne la netteté de l'image : beaucoup d'espaces, sur cette figure, sont blancs. Les distances correspondantes sont proches de 1. Cela correspond à des distances entre des bagages très importantes. De plus, le faible nombre de pavés sombres indique que beaucoup de bagages ont pu être aggrégés ensemble. Pour obtenir les grappes correspondantes, nous devons convenir d'un seuil. La caractéristique "nette" de l'image nous indique que le seuil peut être élevé et que cela n'aura pas une énorme influence sur le nombre de grappes. Nous employons donc un seuil de $0,8$. Ainsi les bagages, pour lesquels la distance deux à deux est inférieure à ce seuil, seront regroupés au sein d'une même grappe. Nous obtenons un ensemble de 17 grappes, présenté dans le tableau 4.26.

Toutes les intrusions au sein d'une même grappe partagent des connaissances. Nous pouvons donc affirmer que ces intrusions sont réalisées par les intrus d'une même communauté. Certaines de ces communautés sont plus actives que d'autres. Le nombre d'intrusions réalisées par la communauté la plus active est 39. Certaines n'en ont réalisé que très peu, comme c'est le cas pour la communauté correspondant à la grappe 8, avec seulement 2 intrusions. Nous sommes à présent capables de savoir à quelle communauté correspond une intrusion. L'analyse du comportement des intrus peut ainsi prendre en compte ce classement pour mener une étude plus fine.

4.6.2 Nature des intrus : êtres humains ou outils automatiques

Avant d'analyser ce que les intrus font sitôt connectés au pot de miel, nous pouvons identifier leur nature. Ils peuvent être de deux natures différentes. Soit ils sont des êtres humains, soit ils sont des outils qui reproduisent des comportements simples. Les êtres humains commettent des erreurs dues par exemple à des manques d'attention, étourderies, défauts de raisonnement ou encore des empressements. En informatique, elles se traduisent souvent par des fautes de frappe. Pour les corriger, l'utilisateur utilise la touche d'effacement arrière, de code ascii 127 (`^?`). Il est vraisemblable que l'observation d'un effacement arrière corresponde à un être humain et non un outil automatique. Nous supposons que les intrus ne cherchent pas à se faire passer pour des outils automatiques. Leur attention est principalement portée sur la manière de mener leur activité sans être démasqué. *Notre premier critère pour identifier les êtres humains est la présence de fautes de frappe dans les commandes saisies.*

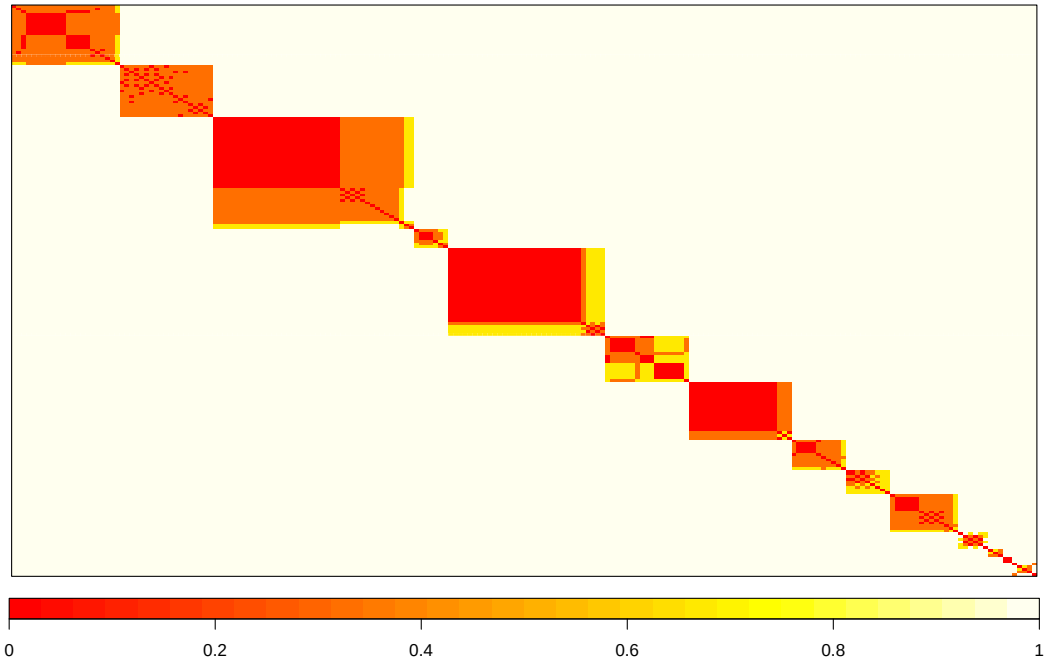


FIG. 4.25 – Distance entre les intrusions

Indice de la grappe	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Nombre d'intrusions	17	7	33	3	39	2	3	2	4	9	19	11	22	2	14	2	21

FIG. 4.26 – Nombre d'intrusions par grappe

Dans certains cas, un être humain peut mener une activité ne contenant que très peu de commandes. Les chances qu'il commette des fautes de frappe sont alors très faibles. De plus, un être humain minutieux peut faire extrêmement attention à la saisie de commandes pour mener une saisie parfaite (sans faute de frappe). Dans ces deux cas, du point de vue du pot de miel, la distinction entre un être humain et un outil automatique est plus subtile. Pour réaliser cette distinction, nous considérons la manière dont les données sont transmises de la machine de l'intrus au pot de miel. Nous pouvons noter que, pour les connexions `ssh`, la transmission des données entre le client et le serveur est asynchrone. La plupart du temps, les implémentations des clients du protocole `ssh` utilisent la fonction `select()` pour obtenir les saisies de l'utilisateur. Lorsque l'utilisateur presse une touche, cette fonction prend fin et le client envoie le caractère correspondant au serveur. Notons que cette manière de procéder est nécessaire pour obtenir des implémentations réactives. En contre partie, des attaques

statistiques existent pour inférer les touches pressées par l'utilisateur[SWT01]. La situation est différente dans le cas d'un "copier-coller" dans le terminal de l'utilisateur client. La fonction `select()` prend aussi fin mais le client ne se cantonne pas à envoyer les données caractère par caractère. Il envoie un tampon contenant l'intégralité des données issues du "copier-coller". Ces transferts de données sont interceptés par la modification apportée à la fonction `tty_read()` (cf. section 3.5.1). Nous admettons que lorsque cette fonction retourne plus d'un caractère, les données ont été transférées par "copier-coller". *Ainsi, une intrusion sans faute de frappe est réalisée par un outil automatique si tous les échanges de données entre le client et le serveur se font bloc par bloc. Cette implication constitue notre second critère.* Notons que, pour les intrusions qui contiennent des transferts caractère par caractère, la présence de transferts bloc par bloc est vraisemblablement due à des raccourcis clavier (tel que la séquence de codes ascii `^V`).

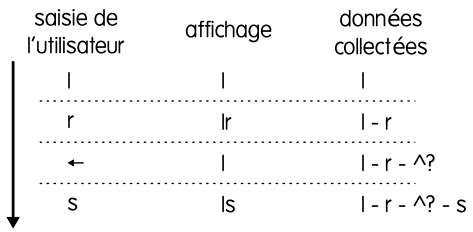


FIG. 4.27 – Exemple de saisie avec une faute de frappe

Au niveau du pot de miel, les saisies de caractères des intrus sont interceptées au niveau de la fonction `tty_read()`. Ces saisies sont datées. Elles constituent les données auxquelles nous prêtons attention dans la suite. Plus précisément, nous regroupons les saisies de caractères par intrusion et nous les ordonnons par date. Nous sommes en mesure de reconstituer ce que saisit et voit l'intrus sur son terminal. L'exemple de la figure 4.27 présente les données collectées au niveau de cette fonction si l'utilisateur saisit la commande `ls` avec une faute de frappe. Il saisit le caractère `r` au lieu de `s` et il utilise le caractère `^?` pour corriger son erreur.

L'application de ces deux critères sur les données nous a permis de classer les intrusions. Les résultats sont présentés sur la figure 4.28. Ainsi, 106 des 210 intrusions contiennent des fautes de frappe. Ces 106 intrusions ont donc été réalisées par des êtres humains. Pour les 104 intrusions réalisées sans faute de frappe, 47 d'entre elles ont été menées caractère par caractère. La nature des intrus à leur origine est vraisemblablement humaine. Quant aux autres intrusions, les activités correspondantes ne sont pas assez importantes pour nous permettre de conclure sur la nature des intrus à leur origine. Effectivement, elles contiennent uniquement une seule commande d'un caractère, telle que `w`. Ainsi, plus des trois quarts des intrusions ont été réalisées par des êtres humains. Le choix de la vulnérabilité proposée aux attaquants (couples (nom d'utilisateur, mot de passe) simples) est donc pertinent.

4.6.3 Les activités des intrus

Dans cette section, nous analysons les commandes lancées par les intrus dans le but d'identifier leurs objectifs. Elle considère les 210 sessions. La première remarque

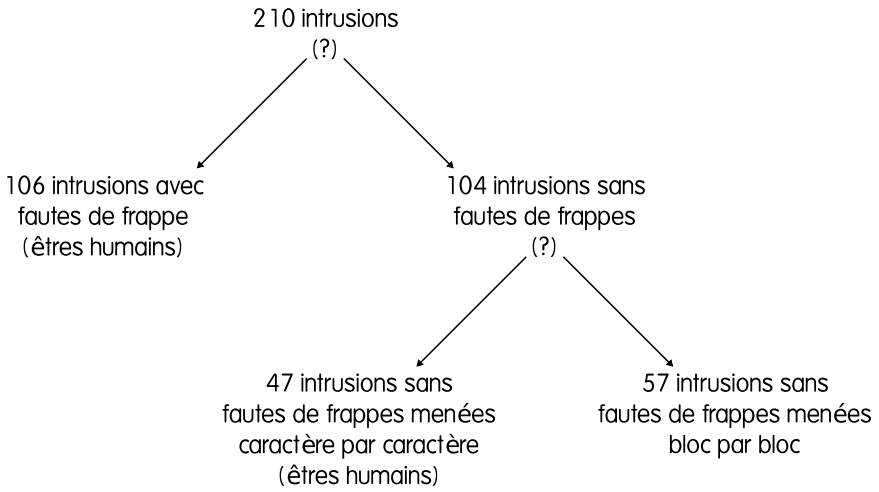


FIG. 4.28 – Nature des intrus

significative est que tous les intrus débutent leur activité par le changement du mot de passe des comptes qu'ils ont découverts. Changer le mot de passe leur permet d'obtenir l'exclusivité du compte pour accéder à la machine. Cette pratique est loin d'assurer la furtivité des intrus : un administrateur système sait si une machine a été piratée. La seconde remarque est que la plupart d'entre eux continuent par le téléchargement de fichiers sur Internet.

La commande la plus souvent utilisée pour le téléchargement est `wget`. Ainsi, 96 des 210 intrusions l'ont employée. Rappelons que les connexions sortantes sont bloquées ou redirigées. Dans tous les cas, une connexion vers un serveur `WEB` ne peut aboutir. Les intrus ont tout de même la possibilité d'initier des connexions dans l'autre sens. Ils peuvent donc télécharger leur fichier sur le pot de miel via la commande `ssh`, et ce, depuis Internet. Il est intéressant d'analyser le pourcentage d'intrusions pour lesquelles les intrus ont réussi à contourner le problème de téléchargement. Aussi surprenant que cela puisse paraître, seulement 30% des intrusions ont utilisé des connexions `ssh` inversées. Autrement dit, 70% des intrusions n'ont pas pu contourner ce problème et ainsi poursuivre leur activité. Trois explications nous semblent pertinentes. Pour la première explication, les intrus suivent des "recettes de cuisine", disponibles sur Internet. Ces recettes expliquent des méthodes d'attaques. La plupart du temps, elles emploient la commande `wget` pour le téléchargement de fichiers. Les intrus, qui ont appliqué ces recettes mais qui n'ont pas réussi à contourner le problème de téléchargement, n'ont pas forcément connaissance d'autres moyens de téléchargement de fichiers. De plus, les intrus qui emploient ces recettes réalisent la plupart du temps un "coller" de l'intégralité de la recette dans son terminal. Pour la seconde explication, nous supposons que le serveur sur lequel réside le fichier à télécharger ne supporte pas les connexions avec la commande `ssh`. Pour la dernière explication, l'intrus se doute de la supercherie en constatant l'impossibilité de contacter des serveurs `WEB` sur Internet et ne poursuit pas son activité par prudence. Etonnamment, la première explication semble être la plus adaptée. Nous avons constaté que plusieurs intrus échouent au téléchargement via la commande `wget` mais reviennent plusieurs jours plus tard en tentant à nouveau le téléchargement, avec la même commande. Certains de ces intrus sont même reve-

nus plusieurs fois. *Les concernant, nous pouvons affirmer deux points. Tout d'abord, ils ne sont pas capables de comprendre pourquoi ce téléchargement échoue. Ensuite, le problème de téléchargement (bloquage des connexions réalisées par la commande* `wget`*) n'altère pas l'intérêt des intrus pour leur cible, notre pot de miel.*

Dès le téléchargement de l'outil réussi, pour 41 des intrusions, les intrus poursuivent par la décompression de ce fichier sur leur cible. Pour approximativement 75% des intrusions ayant réussi le téléchargement, la décompression ne s'effectue pas dans le répertoire du compte courant mais dans des répertoires standards, dans lesquels l'écriture est possible, tels que `/tmp`, `/var/tmp` ou encore `/dev/shm`. Cette démarche est employée par les intrus car elle rend difficile l'identification du type d'activité malveillante. Effectivement, ces répertoires sont couramment employés par les utilisateurs et le système d'exploitation. Quasiment tous les intrus ont décompressé leurs fichiers dans des répertoires cachés. Un repertoire caché sur les systèmes `Unix` est un répertoire dont le nom commence par un point. Donc, bien qu'ils ne se soucient pas de passer inaperçus (changement du mot de passe d'un compte), ils désirent tout de même ne pas dévoiler le type d'activité qu'ils mènent au cœur du système. Les fichiers compressés contiennent des outils d'attaques. En observant leur fonctionnement (par des techniques de désassemblage, par exemple), nous avons identifié plusieurs activités différentes.

Nous avons identifié quatre activités principales, menées par les intrus. La première est le lancement d'un balayage (en anglais, scan) afin d'identifier des machines disposant du service `ssh`. Aucun de ces balayages n'a testé le réseau auquel appartient le pot de miel. Nous supposons que l'intrus a déjà identifié les machines du réseau du pot de miel et disposant du service `ssh`. Après tout, ces intrus ont eu vent de la présence du service `ssh` sur notre machine par le biais d'un balayage du réseau de notre machine. Une seconde hypothèse concerne la prudence d'un intrus. Afin de ne pas éveiller les soupçons, ce dernier peut vouloir minimiser le nombre de machines d'un même réseau qu'il pénètre. L'idée principale est d'utiliser une machine, en l'occurrence notre pot de miel, comme tremplin afin de balayer d'autres réseaux sur Internet. Elle met en évidence le phénomène de rebond. En employant des rebonds, l'intrus brouille les pistes permettant de le localiser. Il masque son identité. Quant aux outils de balayage utilisés, nous pouvons facilement les obtenir sur Internet. L'un d'eux se nomme `pscan.c`. Les connexions initiées depuis le pot de miel et à destination d'Internet sont bloquées à l'exception des connexions à destination du service `ssh`. Ces dernières sont traitées par le mécanisme de redirection. Nous les étudions dans la suite, à la section 4.6.5.

La seconde activité consiste à lancer un client `irc`. Des exemples de client `irc` sont `emech`[Ene] et `psyBNC`. Ces clients ne sont pas des outils d'attaque. Mais leur utilisation a été déviée de leur but initial par les intrus. Ces derniers les utilisent souvent pour permettre aux machines d'un *botnet* de communiquer. Le nom des exécutables a souvent été changé. Sachant que ce client doit être exécuté le plus longtemps possible, ce changement est probablement une tentative de dissimulation de l'activité. Cette attitude coïncide avec leur désir de masquer leur activité sans se soucier d'être furtif. Aujourd'hui, bon nombre d'articles traite de la furtivité des programmes malveillants. Des techniques pointues ont été mises au point. Mais elles ne sont pas employées par les intrus que nous avons observés. Globalement, cette activité est liée à l'intention des intrus d'augmenter la taille de leur *botnet*. A ce niveau, nous supposons que l'intrus

est animé de motivations économiques.

La troisième activité concerne l'augmentation des privilèges. Certains intrus tentent de voler les privilèges de l'administrateur. Etonnamment, nous avons observé seulement 3 intrusions ayant réalisé cette activité. Deux logiciels malveillants différents ont été utilisés. Le premier exploite deux vulnérabilités : une vulnérabilité qui concerne le gestionnaire de mémoire de l'appel système `mremap`[CERb] et une autre qui concerne le gestionnaire chargé de gérer les tas des processus[CERa]. Ces exploits ne peuvent pas fonctionner sur le pot de miel en raison d'un conflit de version. L'intrus aurait dû s'en rendre compte, étant donné qu'il s'est intéressé à la version du système d'exploitation (commande `uname -a`). Pourtant, il a tout de même exécuté son logiciel malveillant. Le test ne l'en a pas dissuadé. Bien entendu, cette exécution a échoué. Le second logiciel malveillant exploite une vulnérabilité dans le programme `ld`. Cet exploit a été utilisé par 3 intrus, qui ont ainsi obtenu un accès en tant que `root` sur le pot de miel. Seulement, ce logiciel n'a réussi que partiellement : par exemple, la suppression de fichiers par l'intrus entraîne des erreurs. En somme, aucun intrus n'a pu exécuter un programme en tant que *root*.

La dernière activité observée sur le pot de miel concerne le phishing. Un seul intrus a réalisé une telle activité. Il a téléchargé un mail pré-rédigé et a essayé de l'envoyer au travers du serveur de mail local (`smtp`). Nous pensons que cette activité ne constitue qu'une première phase de l'attaque car la liste des destinataires du message était courte. Les activités de *phishing* permettent de monter des escroqueries. Elles sont liées, elles aussi, à des motivations économiques.

4.6.4 Compétences des intrus

Les intrus peuvent être classés en deux catégories. La plus importante des deux est celle des *script kiddies*. Ce sont des intrus inexpérimentés. Ils utilisent des programmes obtenus sur Internet sans vraiment comprendre leur fonctionnement. La seconde catégorie est celle des *black hat*. Ils peuvent générer de sérieux dégâts sur les systèmes. Ils sont experts en sécurité et savent comment fonctionnent les outils qu'ils utilisent.

Comme mentionné précédemment, nous avons observé des intrus qui ne sont pas aussi doués que nous le pensions. Par exemple, les activités de certains intrus nous porte à croire qu'ils ne maîtrisent pas le principe des droits d'accès des fichiers sur les système UNIX (ils tentent d'effacer des fichiers qu'il leur est impossible de modifier). Ces mêmes intrus ont aussi tenté de stopper des processus appartenant à d'autres utilisateurs. De plus, une bonne partie des intrus n'a pas caché son activité en effaçant les fichiers d'historiques (`bash_history`, ...).

Une remarque importante à noter sur ces intrus concerne l'identification du pot de miel. Beaucoup d'entre eux ont cherché à obtenir des informations sur le matériel (fichier `/proc/cpuinfo`). Mais aucun n'a utilisé des techniques connues permettant d'identifier la présence des outils VMWARE et QEMU. Vraisemblablement, les intrus que nous avons observés ne sont pas des intrus expérimentés.

De la même manière que pour les attaques par dictionnaire, nous avons confronté la liste des adresses ayant réalisé des intrusions à la liste des adresses observées sur les pots de miel basse interaction de la plate-forme LEURRE.COM. Aucune des adresses ayant réalisé des intrusions n'a été observée sur les pots de miel basse interaction. Les machines utilisées par les attaquants pour réaliser des intrusions semblent aussi être

```
james@M1:~/rep_hacker$ ./unix  66..
[+][+][+][+][+] UnixCoD Atack Scanner [+][+][+][+][+]
[+]   SSH Brute force scanner : user & password   [+]
[+]        Undernet Channel : #UnixCoD            [+]
[+][+][+][+][+][+][+] ver 0x10  [+][+][+][+][+][+][+]
[+] Scanam: 66.221.4.* (total: 2) (1.6% done)
66.221.8.* (total: 2) (3.1% done)
66.221.12.* (total: 2) (4.3% done)
66.221.16.* (total: 2) (5.9% done)
66.221.19.* (total: 2) (7.5% done)
66.221.23.* (total: 2) (9.0% done)
66.221.27.* (total: 2) (10.2% done)
66.221.30.* (total: 2) (11.8% done)
66.221.34.* (total: 2) (13.3% done)
66.221.38.* (total: 2) (14.5% done)
66.221.41.* (total: 2) (16.1% done)
66.221.45.* (total: 2) (17.6% done)
66.221.49.* (total: 2) (18.8% done)
66.221.52.* (total: 2) (20.4% done)
66.221.56.* (total: 2) (22.0% done)
66.221.60.* (total: 2) (23.1% done)
```

Fig. 4.29 – Exemple d'intrusion ayant activé le mécanisme de redirection

spécialisées dans cette activité.

4.6.5 Enseignements apportés par l'emploi du mécanisme de redirection

Rappelons que le mécanisme de redirection que nous avons implémenté et déployé permet de traiter les connexions initiées depuis le pot de miel haute interaction et à destination d'Internet (cf. section 3.5.5 du chapitre 3). Le traitement de ces connexions a nécessité leur interception pouvant entraîner une latence considérable et ainsi alerter les intrus. Des tests ont avant tout été réalisés afin de nous assurer que cette latence est minime pour ainsi prévenir toute tentative de détection[AAN+07b].

Durant la mise en service de ce mécanisme, nous avons pu observer son activation par des intrus. En particulier, les connexions issues d'un balayage de réseaux ont été interceptées et traitées. A ce niveau, nous avons pu constater l'efficacité de ce mécanisme. La figure 4.29 présente un exemple, pour lequel, l'intrus a lancé un balayage à l'encontre du réseau `66.221.*`. A l'issu de ce balayage, l'intrus a pu constater que deux connexions avaient réussi. A ce niveau, il peut croire que deux adresses du réseau `66.221.*` proposent un accès au service `ssh`. En réalité, les connexions à ces deux adresses sont redirigées vers les autres machines de notre pot de miel haute interaction.

Globalement, nous avons constaté que les intrus ayant activé le mécanisme de redirection n'exploitent pas les informations qu'ils viennent de découvrir depuis le pot de miel. A ce niveau, nous pouvons émettre deux hypothèses. Soit ils ont pris conscience de la présence du mécanisme de redirection et stoppé leur activité, soit

ils utilisent ces informations depuis d'autres machines sur Internet. Sachant que les intrus sont revenus après avoir obtenu leurs informations, il est vraisemblable que la deuxième hypothèse soit la plus plausible. Cette constation nous conforte dans l'idée que les attaquants spécialisent les machines, dont ils ont pris le contrôle, pour des tâches particulières.

4.7 Conclusion

Dans ce chapitre, nous avons développé une méthodologie et analysé les données collectées sur le pot de miel haute interaction. 419 jours de collecte de données ont été exploités. Le but de ces analyses est d'étudier le comportement des attaquants. Nous avons mis en évidence le processus d'intrusion qu'ils ont suivi. Ce processus est tout d'abord constitué d'une phase d'attaque par dictionnaire, suivie d'une phase d'intrusion.

Les résultats obtenus sont en accord avec les hypothèses sur les attaquants qui circulent dans les communautés liées à la *sécurité* et les premières expérimentations que nous avons présentés dans [ANK+06]. La contribution de ces travaux consiste à mettre en œuvre un environnement expérimental permettant d'enregistrer les activités des intrus et à mener une analyse rigoureuse pour étudier leur comportement. La figure 4.30 présente une vision globale du processus de traitement ayant permis ces analyses.

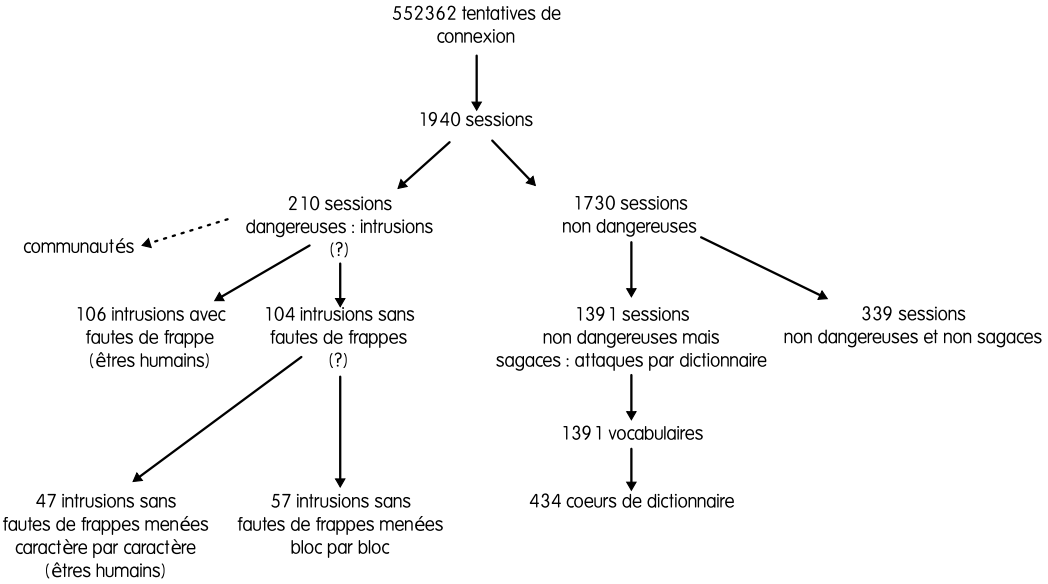


Fig. 4.30 – Processus de traitement et classification des données

Ces analyses révèlent des points intéressants. Tout d'abord, les attaquants affectent un rôle particulier à chaque machine qu'ils utilisent pour attaquer leurs cibles. Certaines sont exclusivement utilisées pour les attaques par dictionnaire et d'autres sont utilisées pour les intrusions. Ensuite, un faible nombre de dictionnaires sources ont été utilisés pour réaliser l'ensemble des attaques par dictionnaire. De plus, ces dictionnaires sources ne sont pas adaptés au système ciblé, de part la langue employée. Nous avons aussi constaté que l'identification de la présence du pot de miel n'est pas

une pratique courante chez les attaquants, contrairement à ce que l'on pourrait croire. Ces attaquants ne sont pas intéressés par masquer leur arrivée (changement des mots de passe) mais plutôt par masquer leur activité.

Durant cette analyse nous avons sûrement levé plus de questions que nous n'avons répondu à des interrogations. A notre connaissance, il n'y a pas d'étude expérimentale équivalente dans la littérature. Les résultats préliminaires de notre expérimentation sont encourageants et nécessitent d'être appronfondis et étendus.

Les résultats de notre étude sont pondérés par les observations que nous avons réalisées sur notre pot de miel. Tout d'abord, il serait intéressant de prolonger ces expériences pour détecter des changements éventuels dans le comportement des attaquants. Il est aussi important de déployer et d'installer le pot de miel en d'autres points d'Internet : les réseaux qui peuvent accueillir le pot de miel ne suscitent pas tous les mêmes intérêts des attaquants. Les résultats obtenus pourraient être liés à la localisation géographique du pot de miel.

Concernant l'implémentation et la configuration de notre pot de miel, nous avons identifié certaines contraintes qui nécessitent d'être traitées pour améliorer le niveau d'observabilité offert par le pot de miel. Nous pouvons citer notamment le problème lié au comportement des attaquants qui ont tendance à changer systématiquement le mot de passe, rendant l'accès difficile à d'autres attaquants. Une solution pour pallier ce problème est actuellement en cours de développement et vise à créer dynamiquement des comptes et des machines virtuelles, sans perturber les processus d'intrusion en cours.

Enfin, il est important d'envisager d'autres vulnérabilités et points d'accès à notre pot de miel. Ceci nous permettrait d'observer des processus d'attaque plus riches afin d'élaborer des modèles plus réalistes. Ces derniers pourraient servir à terme par exemple pour valider le modèle d'évaluation quantitative de la sécurité basé sur le graphe des privilèges développé au Laas-Cnrs[DD94]

Conclusion

Dans cette thèse, nous avons mené des analyses sur le comportement des attaquants en nous basant sur les pots de miel. Dans un premier temps, nous avons exploité les données collectées sur des environnements de pot de miel basse interaction dans le cadre du projet Leurre.com, afin de modéliser le comportement des attaquants sur Internet. Ensuite, nous avons développé et déployé un environnement de pot de miel haute interaction afin d'observer et d'analyser le comportement des attaquants une fois introduits dans le cœur du système. Pour chacune des analyses, nous avons proposé une méthodologie.

L'étude des données collectées dans le cadre du projet Leurre.com a été focalisée sur l'élaboration de modèles stochastiques caractérisant le processus d'occurrence des attaques observées et leur propagation. Nous avons étudié plus particulièrement les distributions des intervalles de temps entre attaques et l'évolution du nombre d'attaques par unité de temps sur différents environnements toutes sources confondues et en fonction de leur origine géographique. Nous avons aussi étudié des corrélations entre les activités observées sur plusieurs environnements, ainsi que les propagations d'attaques entre ces environnements.

Les informations collectées à partir de la plate-forme Leurre.com représentent un volume important de données et ont été enregistrées à partir d'un grand nombre d'environnements de pots de miel qui ont été déployés progressivement pendant la période considérée. Afin d'effectuer des analyses comparatives significatives et de minimiser l'impact des périodes d'indisponibilité qui ont affecté les différents environnements, nous avons développé une méthodologie basée sur des outils statistiques qui permet d'identifier ces périodes d'indisponibilité, et de guider le choix des données utilisées pour élaborer et valider les modèles. Nous avons étudié trois types de modèles qui ont apporté des éclairages complémentaires sur les données analysées :

- La modélisation des intervalles de temps entre attaques observées sur les différents environnements étudiés a permis d'identifier une distribution de probabilité qui met en évidence deux types de trafics malveillants qui cohabitent : les attaques en rafales et des attaques monotones plus espacées dans le temps. Ces modèles peuvent être utilisés pour générer du trafic malveillant représentatif des activités observées sur les pots de miel basse interaction, pour aider par exemple à la validation de mécanismes de protection vis-à-vis des malveillances.
- Les analyses de corrélation concernant l'évolution de la fréquence d'occurrence des attaques en considérant les données observées sur différents environnements ont permis de révéler des similitudes de comportements entre certains environnements alors qu'ils ne sont pas localisés géographiquement au même endroit et ne possèdent pas des adresses IP proches. Ces analyses ont aussi mis en évidence

d'autres comportements surprenants, par exemple concernant la forte corrélation entre l'évolution du trafic malveillant toutes sources confondues et le trafic provenant d'un seul pays qui ne représente qu'une faible proportion de l'activité globale.
- L'analyse des propagations des attaques montre que les intrus ont tendance à visiter de façon successive différents environnements, plutôt que de rester dans le même environnement. L'évaluation des probabilités caractérisant les propagations entre les environnements, permet aussi d'identifier les environnements qui manifestent de fortes dépendances et aussi ceux vers lesquels les propagations sont généralement faibles.

Ces analyses ont permis de caractériser les processus d'attaque à l'entrée des environnements de pots de miel basse interaction. Nous nous sommes aussi intéressés dans le cadre de cette thèse à l'étude du comportement des attaquants une fois introduits dans le cœur du système. A cet effet, nous avons opté pour l'utilisation d'un pot de miel haute interaction. Notre objectif était de proposer aux attaquants une vulnérabilité des plus anciennes, les mots de passe simples, en utilisant le service de connexion à distance `ssh`. Les implémentations proposées dans la littérature n'étant pas adaptées à nos besoins, nous avons développé notre propre implémentation.

Cette implémentation prend en compte deux aspects. Le premier correspond aux techniques mises en œuvre pour collecter des données issues des activités des attaquants. Plus particulièrement, nous avons modifié le noyau du système d'exploitation afin de récupérer les informations nécessaires pour la reconstitution des activités des différents attaquants. Ces informations comprennent les noms d'utilisateur et mots de passe tentés ainsi que les touches pressées par les attaquants. Le deuxième aspect correspond à une technique permettant de contourner une limite liée à l'aspect juridique de l'utilisation des pots de miel : il est interdit de proposer à la communauté des attaquants sur Internet des ressources dans un réseau leur permettant de rebondir et ainsi attaquer d'autres machines sur Internet. La solution que nous avons proposée à cette limite consiste à rediriger les connexions initiées par les attaquants depuis le pot de miel vers d'autres pots de miel, de façon dynamique.

Sur la base des données collectées sur le pot de miel haute interaction, nous avons mis en place une méthodologie d'analyse permettant de caractériser deux types d'attaques : les attaques par dictionnaire et les intrusions. Tout d'abord, nous avons décrit une méthode permettant de regrouper les tentatives de connexions au service `ssh` en session. Une session représente au mieux l'activité d'un attaquant sur le pot de miel haute interaction. Nous avons discerné deux types de sessions différents. Les sessions constituées d'un nombre important de connexions, avec aucune connexion ayant abouti à une saisie de commande, forment les attaques par dictionnaire. Les sessions constituées de connexions ayant abouti à une saisie de commandes forment les intrusions. L'analyse de ces deux types d'attaque a permis de valider des intuitions concernant le comportement des attaquants :
- Les attaques par dictionnaire observées sur le pot de miel ont été réalisées à partir d'un faible nombre de dictionnaires sources seulement. La plupart de ces dictionnaires sources sont constitués de mots des langues anglaise et francaise. Certains noms d'utilisateur sont testés plusieurs fois, avec des mots de passe différents. Ces dictionnaires sources ne sont pas constitués de manière aléatoire. En revanche, en analysant plus en détail le contenu de ces dictionnaires sources,

nous constatons qu'ils ne sont pas adaptés à la langue du pays qui héberge le pot de miel haute interaction.
- Concernant les attaquants ayant réalisé des intrusions (les intrus), nous avons analysé les commandes qu'ils ont réalisées afin de reconstituer leur activité. Nous remarquons que la plupart des attaquants saisissent les commandes caractère par caractère en faisant beaucoup de fautes de frappe. Certains d'entre eux ont saisi les commandes en effectuant des copier-coller des recettes sur Internet et en les envoyant sur la connexion `ssh`. Nous pouvons donc en déduire que ces intrus sont très vraisemblablement des êtres humains.
- Les activités qu'ils ont réalisé suivent un schéma simple, dénoué de toute tentative d'identification du pot de miel. Aucun d'entre eux n'a tenté de vérifier si les actions qu'il voulait réaliser étaient compatibles avec la version des outils disponibles sur le pot de miel.
- Les intrus que nous avons observés ne sont pas des experts informatiques.
- De plus, les intrusions ont été réalisées en s'appuyant sur les informations collectées par les attaquants lors des attaques par dictionnaire et plusieurs intrusions réalisées par des adresses différentes ont utilisé les mêmes connaissances sur le pot de miel haute interaction. Il est vraisemblable que les attaquants sur Internet se regroupent en communautés pour mener leurs activités.
- Aussi bien pour les attaques par dictionnaire que pour les intrusions, nous avons observé que les adresses des attaquants n'ont jamais été observées sur les pots de miel basse interaction. Ceci nous amène à supposer que les attaquants spécialisent les machines qu'ils utilisent pour leurs attaques.

Les possibilités offertes par les données collectées sur les pots de miel sont vastes. Aussi, il nous semble important que ces analyses se poursuivent afin d'enrichir les connaissances acquises sur le comportement des attaquants. Dans cette optique, nous avons identifié plusieurs axes de recherche.

Tout d'abord, nous pourrions considérer des modèles plus complexes pour la modélisation du trafic malveillant avec par exemple des chaînes de Markov cachées dans le but de mieux capturer toutes les particularités du trafic malveillant. De plus, d'autres modèles pourraient constituer de bonnes alternatives dans le but d'évaluer de façon prévisionnelle les activités malveillantes sur Internet et d'utiliser ces prévisions pour améliorer l'efficacité des mécanismes de detection d'intrusion.

Proposer de nouvelles vulnérabilités aux attaquants sur le pot de miel haute interaction ou installer des pots de miel haute interaction avec d'autres systèmes d'exploitation hôte pourraient permettre de diversifier les communautés observées, c'est-à-dire de ne pas se limiter à l'observation des communautés qui s'introduisent au cœur des systèmes par le biais d'attaques par dictionnaire sur le service `ssh`. L'installation et l'instrumentation de systèmes d'exploitation `Windows`, ainsi que l'ouverture de vulnérabilités typiques de ce système, paraît donc une perspective naturelle à ces travaux.

Les analyses concernant le comportement des intrus peuvent être utilisées comme base de départ pour valider à long terme l'approche d'évaluation quantitative de la sécurité informatique développée au Laas[DD94] (cf. section 1.3.1 du chapitre 1) en confrontant les observations aux différentes hypothèses du graphe des privilèges. En effet, le graphe des privilèges permet d'identifier de façon théorique différents parcours d'un attaquant un sein d'un ensemble de machines compromises. Ces parcours sont obtenus en faisant des hypothèses sur le comportement des attaquants. L'observation

de ces attaquants à l'aide de pots de miel haute interaction massivement répartis nous permettrait de valider ces hypothèses théoriques en observant réellement la façon dont les intrus se comportent.

Enfin, il reste des améliorations à apporter au niveau de l'implémentation du pot de miel haute interaction proposée. Le mécanisme de rebonds que nous avons implémenté a montré son efficacité mais aussi ses limites. Imaginer des améliorations à ce mécanisme nous semble également un travail futur intéressant. De la même façon, il nous semble pertinent des réfléchir à des solutions permettant d'attirer des intrus expérimentés sur nos systèmes. Même s'ils ne représentent pas une énorme proportion du nombre d'attaquants, leurs techniques d'attaque, particulièrement dangereuses, méritent d'être étudiées de façon à en tirer de précieux enseignements.

Table des figures

1.1 La sûreté de fonctionnement 9
1.2 Méthodes disponibles pour la mise en œuvre des moyens de la sûreté de fonctionnement 11

2.1 Architecture du pot de miel basse interaction, projet Leurre.com . 27
2.2 Evolution du nombre d'environnements déployés sur Leurré.com et durée d'activité pour chacun 30
2.3 Nombre d'adresses, en ordonnée, ayant réalisé le nombre de sessions correspondant 31
2.4 Liste des adresses ayant réalisé les plus fortes activités 31
2.5 Evolution du nombre de sessions observées par jour sur l'environnement 37 32
2.6 Variables représentant les dates des sessions et les durées entre deux sessions consécutives 35
2.7 Processus de traitement des données collectées sur les environnements de pot de miel déployés sur Internet 36
2.8 Exemple de *boxplot* 37
2.9 Exemples de distributions 38
2.10 Evolution du nombre de sessions par jour pour certains environnements 42
2.11 Evolution du nombre de jours pour la sous-période, en fonction du nombre de périodes de silence suspectes et du nombre d'environnements minimum 46
2.12 Test du χ^2 51
2.13 Densités de probabilité des environnements 9, 13, 14 et 28 52
2.14 Ajustement du modèle de mélange et comparaison avec d'autres modèles 55
2.15 Ajustement du modèle de mélange, échelles logarithmiques 55
2.16 Evolution du nombre de sessions toutes origines confondues et évolution du nombre de sessions provenant des Etats-Unis, pour l'environnement 9 57
2.17 Evolution du nombre de sessions observé et du nombre de sessions estimé par le modèle de régression, pour tous les environnements . . . 60
2.18 Exemple de propagation d'un ver 62
2.19 Graphe de propagation associé aux 8 environnements 65

3.1 Architecture d'un pot de miel basé sur Prelude et Uml 70
3.2 Mécanisme d'interception d'un appel système 71
3.3 L'échange de clé Diffie-Hellman 75
3.4 Méthode d'inférence du contenu de connexions chiffrées 76

3.5 Le principe de redirection à la volée 78
3.6 Architecture générale du pot de miel haute interaction 79
3.7 Implémentation du pot de miel haute interaction 83
3.8 Architecture du mécanisme de redirection des connexions 86
3.9 Configuration du premier déploiement 87
3.10 Configuration du deuxième déploiement 87

4.1 Liste des couples les plus employés 90
4.2 Évolution de la durée entre les connexions successives 91
4.3 Fréquences des durées entre connexions successives 91
4.4 Lien entre les valeurs τ_1 et τ_2 92
4.5 Temps entre les premières connexions pour chaque couple déclaré . . 92
4.6 Impression sur une feuille. 95
4.7 Exemple d'application de l'algorithme de la fenêtre glissante 95
4.8 Exemple de construction d'ensemble de sessions 97
4.9 Cas de figure différents pour la détermination du seuil de regroupement. 98
4.10 Evolution du nombre de sessions en fonction du seuil 99
4.11 Vitesse de regroupement des sessions 99
4.12 Classification des sessions 101
4.13 Caractéristiques statistiques des tailles des vocabulaires 104
4.14 Rapport du nombre de couples différents répétés sur le nombre de couple différents pour chaque vocabulaire 105
4.15 Exemple de couverture de vocabulaires 106
4.16 Représentation des ensembles utilisés dans le calcul de la distance intertextuelle 107
4.17 Evolution de la distance en fonction de la couverture et de la ressemblance 109
4.18 Exemple de partitionnement 110
4.19 Exemple de représentation matricielle 111
4.20 Distances entre les vocabulaires 113
4.21 Exemple de construction du cœur de dictionnaire pour une grappe . . 114
4.22 Profils des cœurs de dictionnaire 115
4.23 Nombre d'intrusions par compte 117
4.24 Evolution des connaissances de l'intrus 118
4.25 Distance entre les intrusions 120
4.26 Nombre d'intrusions par grappe 120
4.27 Exemple de saisie avec une faute de frappe 121
4.28 Nature des intrus 122
4.29 Exemple d'intrusion ayant activé le mécanisme de redirection 125
4.30 Processus de traitement et classification des données 126

Bibliographie

[AAN+07a] E. Alata, I. Alberdi, V. Nicomette, P. Owezarski et M. Kaâniche : Mécanisme d'observation d'attaques sur internet avec rebonds. *Dans Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC'07)*, pp. 53–67, 2007. Adresse Internet : `actes.sstic.org/SSTIC07/Observation_Attaques_Internet_Rebond`.

[AAN+07b] I. Alberdi, E. Alata, V. Nicomette, P. Owezarski et M. Kaâniche : Shark : Spy Honeypot with Advanced Redirection Kit. *Dans IEEE Workshop on Monitoring, Attack Detection and Mitigation (MonAM'07)*, pp. 47–52, Toulouse, France, 2007.

[ACK+03] J. Arlat, Y. Crouzet, J. Karlsson, P. Folkesson, E. Fuchs et G.H. Leber : Comparison of Physical and Software-Implemented Fault Injection Techniques. *IEEE Transactions on Computers*, vol. 52, num. 9, pp. 1115–1133, 2003, Washington, DC, USA.

[Adl90] L.M. Adleman : *An Abstract Theory of Computer Viruses*, pp. 354–374. Springer-Verlag, London, UK, 1990. isbn : 3-540-97196-3.

[AGJ05] I. Alberdi, J. Gabès et E. Le Jamtel : UberLogger : un observatoire niveau noyau pour la lutte informatique défensive. *Dans Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC'05)*, 2005. Adresse Internet : `actes.sstic.org/SSTIC05/UbberLogger`.

[AH07] X. Allamigeon et C. Hymans : Analyse statique par interprétation abstraite. *Dans Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC'07)*, 2007. Adresse Internet : `actes.sstic.org/SSTIC07/Heap_Overflow_Analyse_Statique`.

[ALRL04] A. Avizienis, J.C. Laprie, B. Randell et C. Landwehr : Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable and Secur. Comput.*, vol. 1, num. 1, pp. 11–33, 2004, Los Alamitos, CA, USA.

[ANK+06] E. Alata, V. Nicomette, M. Kaâniche, M. Dacier et M. Herrb : Lessons learned from the deployment of a high-interaction honeypot. *Dans 6th European Dependable Computing Conference (EDCC'06)*, pp. 39–44, Coimbra, Portugal, 2006.

[ASS+06] H. Artail, H. Safa, M. Sraj, I. Kuwatly et Z. Al Masri : A hybrid honeypot framework for improving intrusion detection systems in protecting organizational networks. *Computers & Security*, vol. 25, num. 4, pp. 274–288, 2006.

[Ban07] D. Bancal : Un pirate informatique annonce avoir piégé un bookmaker britannique. Adresse Internet : www.zataz.com/alerte-phishing/14295/casino-bookmarker-argent.html, 2007.

[Bar64] P. Baran : On distributed communications networks. *IEEE Transactions on Communications*, vol. CS-12, num. 1, mars 1964.

[BCW+04] M. Bailey, E. Cooke, D. Watson, F. Jahanian et N. Provos : A Hybrid Honeypot Architecture for Scalable Network Monitoring. Rapport technique CSE-TR-499-04, University of Michigan, USA, Ann Arbor, MI, USA, 2004.

[Bel] F. Bellard : Qemu. Adresse Internet : fabrice.bellard.free.fr.

[Bel92] S.M. Bellovin : There be dragons. *Dans Proceedings of the Third Usenix UNIX Security Symposium*, pp. 1–6, Baltimore, MD, USA, 1992.

[Ber86] R. Bertrand : *Pratique de l'analyse statistique des données*. Presses Universitaires du Québec, 1986. isbn : 2-7605-0382-8.

[Ber03] S. Berinato : How a Bookmaker and a Whiz Kid Took On an Extortionist – and Won. *CSO*, 2003. Adresse Internet : www.csoonline.com/read/050105/extortion.html.

[BHS04] G. Brys, M. Hubert et A. Struyf : A robust measure of skewness. *Journal of Computational & Graphical Statistics*, vol. 13, pp. 996–1017, 2004.

[Bil97] J. Bilmes : A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Rapport technique ICSI-TR-97-021, University of Berkeley, 1997.

[BKH+06] P. Baecher, M. Koetter, T. Holz, M. Dornseif et F. Freiling : The Nepenthes Platform : An Efficient Approach to Collect Malware. *Dans 9th International Symposium On Recent Advances In Intrusion Detection (RAID'06)*, Lecture Notes in Computer Science 4219, pp. 165–184, Hambourg, Allemagne, septembre 2006. Springer-Verlag.

[BL89] T. Berners-Lee : Information management : A proposal. www.w3.org/History/1989/proposal.html, 1989.

[BLOJ94] S. Brocklehurst, B. Littlewood, T. Olovsson et E. Jonsson : On measurement of Operational Security. *Aerospace and Electronic Systems Magazine, IEEE*, vol. 9, num. 10, pp. 7–16, 1994.

[BSC+07] A. Neves Bessani, P. Sousa, M. Correia, N. Ferreira Neves et P. Verissimo : Intrusion-Tolerant Protection for Critical Infrastructures. DI/FCUL TR 07–8, Department of Informatics, University of Lisbon, 2007. Adresse Internet : www.di.fc.ul.pt/tech-reports/07-8.pdf.

[BSHN04] R. Budiarto, A. Samsudin, C. Wee Heong et S. Noori : Honeypots : Why We Need a Dynamic Honeypots ? *Dans Proceedings of the 1st International Conference on Information and Communication Technologies : from Theory to Applications (ICTTA'04)*, pp. 565, Damas, Syrie, 2004.

[BUG] BUGtrack – Web Based Bug Tracking and Project Management Software. Adresse Internet : www.bugtrack.net.

[Cab07] A. Cabezon : Un pirate britannique arrêté, 2007. Adresse Internet : www.vulnerabilite.com/pirate-mckinnon-actualite-20050608165503.html.

[CBFF03] B. Caswell, J. Beale, J.C. Foster et J. Faircloth : *Snort 2.0 Intrusion Detection*. Syngress, 2003. isbn : 1931836744.

[CC06] F. Cuppens et N. Cuppens : *Les modèles de sécurité*, chapitre Sécurité des systèmes d'information, (Traité IC2) sous la direction de Yves Deswarte et Ludovic Mé, pp. 13–48. Série Réseaux et télécoms. Hermès - Lavoisier, 2006.

[CC999] ISO, Common Criteria for Information Technology Security Evaluation, Part1 : Introduction and General Model. Norme ISO/IEC 15408, 1999.

[CCS] The official website of the common criteria project. Adresse Internet : www.commoncriteriaportal.org.

[CERa] CERT : Linux kernel do_brk() function contains integer overflow. Adresse Internet : www.kb.cert.org/vuls/id/981222.

[CERb] CERT : Linux kernel mremap(2) system call does not propertly check return value from do_munmap() function. Adresse Internet : www.kb.cert.org/vuls/id/981222.

[CER07] CERT Coordination Center : CERT/CC statistics 1988-2007, 2007. Adresse Internet : www.cert.org/stats.

[Che91] B. Cheswick : An Evening with Berferd in Which a Cracker is Lured, Endured, and Studied. *Dans Proceedings of the Winter 1992 USENIX Conference*, pp. 163–174, San Francisco, CA, USA, 1991.

[Cla01] K. C. Claffy : Caida : Visualizing the internet. *IEEE Internet Computing*, vol. 5, num. 1, pp. 88, 2001.

[Clu07] Clusif : Cybercrime Overview – 2006. Rapport technique, Clusif, Club de la Sécurité de l'Information Français, 2007. www.clusif.asso.fr.

[CM02] F. Cuppens et A. Miège : Alert correlation in a cooperative intrusion detection framework. *Dans Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pp. 202–215, Washington, DC, USA, 2002. IEEE Computer Society.

[Coh87] F. Cohen : Computer Viruses : Theory and Experiments. *Computers & Security*, vol. 6, num. 1, pp. 22–35, 1987, Oxford, UK.

[Col97] M. Collins : The EM algorithm, 1997. Adresse Internet : people.csail.mit.edu/mcollins/papers/wpeII.4.ps.

[Cor04] J. Corey : Local Honeypot Identification. *Dans Phrack 62*, volume 0x0b, 2004. Adresse Internet : www.phrack.org.

[CR04] T. Chen et J.M. Robert : The evolution of viruses and worms. *Dans Statistical Methods in Computer Security*. CRC Press, 2004.

[CS04] C. Csallner et Y. Smaragdakis : JCrasher : An automatic robustness tester for Java. *Software – Practice & Experience*, vol. 34, num. 11, pp. 1025–1050, 2004.

[CTC93] CSEC, The Canadian Trusted Computer Product Evaluation Criteria. Version 3.0e, Canadian System Security Center, Communications Security Establishment of Canada, 1993.

[CVE] Common Vulnerabilities and Exposures – CVE. Adresse Internet : `cve.mitre.org`.

[Dac94] M. Dacier : *Vers une évaluation quantitative de la sécurité informatique*. Thèse de doctorat, Institut national polytechnique de Toulouse, 1994.

[DBF91] Y. Deswarte, L. Blain et J.C. Fabre : Intrusion Tolerance in Distributed Computing Systems. *Dans Proceedings of the 1991 IEEE Symposium on Security and Privacy*, pp. 110–121, Oakland, CA, USA, 1991.

[DD94] M. Dacier et Y. Deswarte : Privilege Graph : an Extension to the Typed Access Matrix Model. *Dans Proceedings of the Third European Symposium on Research in Computer Security (ESORICS'94)*, pp. 319–334, London, UK, 1994. Springer-Verlag.

[DDW98] H. Debar, M. Dacier et A. Wespi : Reference Audit Information Generation for Intrusion Detection Systems. *Dans* Reinhard Posch et György Papp, éditeurs : *Information Systems Security, Proceedings of the 14th International Information Security Conference (IFIP SEC'98)*, pp. 405–417, 1998. Vienne, Autriche et Budapest, Hongrie.

[Del02] F. Dellaert : The Expectation Maximization Algorithm. Rapport technique GIT-GVU-02-20, College of Computing, Georgia Institute of Technology, 2002. Adresse Internet : `www.cc.gatech.edu/~dellaert/em-paper.pdf`.

[Des03] Y. Deswarte : Comment peut-on tolérer les intrusions sur internet ? : Les systèmes critiques face aux malveillances. *La Revue de l'Electricité et de l'Electronique*, vol. 8, pp. 83–90, 2003.

[DHK04] M. Dornseif, T. Holz et C. Klein : NoSEBrEaK - attacking honeynets. *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pp. 123–129, juin 2004.

[Dik06] J. Dike : *User Mode Linux*. Prentice Hall Ptr, 1st édition, 2006. isbn : 0-13-186505-6.

[DJG07] F. Dressler, W. Jaegers et R. German : Flow-based Worm Detection using Correlated Honeypot Logs. *Dans* Torsten Braun and Georg Carle and Burkhard Stiller, éditeur : *15. Gi/Itg Fachtagung Kommunikation in Verteilten Systemen (KiVS'2007)*, pp. 181–186, Bern, Suisse, 2007. iVde.

[DKD93] M. Dacier, M. Kaâniche et Y. Deswarte : A framework for security assessment of insecure systems. First Year Report of the ESPRIT Basic Research Action 6362 : Predictably Dependable Computing Systems (PDCS2), 1993.

[DLR77] A. Dempster, N. Laird et D. Rubin : Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, vol. 39, num. Series B, pp. 1–38, 1977.

[DMC06] D. Duncombe, G. Mohay et A. Clark : Synapse : auto-correlation and dynamic attack redirection in an immunologically-inspired ids. *Dans Proceedings of the 2006 Australasian workshops on Grid computing and e-research (ACSW Frontiers'06)*, pp. 135–144, Hobart, Tasmanie, Australie, 2006. Australian Computer Society, Inc. isbn : 1-920-68236-8.

[DMD06] M. Dacier, L. Mé et Y. Deswarte : *Sécurité des systèmes d'information (Traité IC2, série Réseaux et télécoms)*, chapitre Détection d'intrusions : état de l'art, faiblesses et problèmes ouverts. Lavoisier, 2006.

[DP06] Y. Deswarte et D. Powell : Internet security : An intrusion-tolerance approach. *Proceedings of the IEEE*, vol. 94, num. 2, pp. 432–441, 2006.

[DS90] D.Farmer et E.H. Spafford : The COPS security checker system. *Dans USENIX Summer*, pp. 165–170, 1990.

[DSH] DShield, Cooperative Network Security Community, Internet Security. Adresse Internet : `www.dshield.org`.

[Ene] EnergyMech team : EnergyMech. Adresse Internet : `www.energymech.net`.

[ETR] eTrust network protection software. Adresse Internet : `www.network-protection.com`.

[Fil04] E. Filiol : L'évolution des idées en virologie informatique. *Dans 7ème Colloque d'Histoire de l'Informatique et de télécommunications Rennes (CHIR'2004)*, pp. 155–170, 2004.

[Fla05] H. Flake : Analyse differentielle de binaires. *Dans Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC'05)*, 2005. Adresse Internet : `actes.sstic.org/SSTIC05/Analyse_differentielle_de_binaires`.

[GD98] S. Grundschober et M. Dacier : Design and Implementation of a Sniffer detector. *Dans Proceedsngs of the First International Workshop on the Recent Advances in Intrusion Detection (RAID'98)*, Louvain-la-Neuve, Belgique, 1998.

[GdRdTd] Gestionnaire du Réseau de Transport d'Electricité : La panne électrique du 4 novembre 2006, questions/réponses. Adresse Internet : `www.rte-france.com/htm/fr/accueil/coupure.jsp`.

[GLR+03] V. Gupta, V. Lam, H. Ramasamy, W. Sanders et S. Singh : Dependability and Performance Evaluation of Intrusion-Tolerant Server Architectures. *Dans Dependable Computing : Proceedings of the First Latin-American Symposium (LADC'2003)*, pp. 81–101, 2003.

[Gru] S. Grundschober : Sniffer detector report. IBM Research Division – Zurich Research Laboratory ; Global Security Analysis Lab.

[Gru69] F. Grubbs : Procedures for detecting outlying observations in samples. *Technometrics*, vol. 11, num. 1, pp. 1–21, 1969.

[Gél] J. Gélinas : Linux-VServer. Adresse Internet : `linux-vserver.org`.

[HA04] V. Hodge et J. Austin : A Survey of Outlier Detection Methodologies. *Artif. Intell. Rev.*, vol. 22, num. 2, pp. 85–126, 2004, Norwell, MA, USA.

[HB03] E. Haugh et M. Bishop : Testing C programs for buffer overflow vulnerabilities. *Dans Proceedings of the Network and Distributed System Security Symposium (NDSS'03)*, San Diego, CA, USA, 2003. The Internet Society.

[HM03] M. Hervieux et T. Meurisse : Uml comme pot à miel. *Dans Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC'03)*, 2003. Adresse Internet : `actes.sstic.org/SSTIC03/Honeypots_UML`.

[Hon03] Honeynet Project : Know Your Enemy : Learning with VMware, Building Virutal Honeynets using VMware, 2003. Adresse Internet : `www.honeynet.org`.

[Hum] J.P. Humbert : Enquête « cybercrime ». Adresse Internet : `www.cases.public.lu/publications/recherche/these_jph/enquete/index.html`.

[Hum06] J.P. Humbert : Les mondes de la cyber-délinquance et images sociales du pirate informatique. Rencontre SPIRAL 2006-09-26 - S'il te plaît... dessine moi un pirate informatique, 2006. Adresse Internet : `www.cases.public.lu/fr/publications/presentations/2006-09-26-Spiral/2_Humbert/index.html`.

[HV06] M. Hubert et E. Vanderviere : An adjusted boxplot for skewed distributions. Rapport technique TR-06-11, Université catholique de Louvain, 2006.

[ISC] Internet Systems Consortium, ISC. Organisme public d'aide à la mise en œuvre d'un " Internet universel auto-organisé ", Adresse Internet : `www.isc.org`.

[ITS91] CCE, Critères d'évaluation de la sécurité des Systèmes informatiques. Commission des Communautés Européenne, 1991.

[IWS] Internet World Stats – Usage and Population Statistics. Adresse Internet : `www.internetworldstats.com`.

[Jac07] P. Jackson : Les cyberpirates prennent l'Estonie à l'abordage. *Courrier International - Bbc News Online*, 2007.

[JCS92] MITT, The Japanese Computer Security Evaluation Criteria – Functionality Requirements. Draft V1.0. Ministry of International Trade and Industry, 1992.

[JNO03] S. Jajodia, S. Noel et B. O'Berry : Topological analysis of network attack vulnerability. *Dans Managing Cyber Threats : Issues, Approaches and Challenges*, 2003.

[Kle61] L. Kleinrock : *Information Flow in Large Communication Nets*. Thèse de doctorat, Massachusetts Institute of Technology, 1961.

[Kor] K. Kortchinsky : Multiple patch for VMware. Adresse Internet : `honeynet.rstack.org/tools/vmpatch.c`.

[KWBS] G. Kuenning, P. Willisson, W. Buehring et K. Stevens : International Ispell. Adresse Internet : `lasr.cs.ucla.edu/geoff/ispell.html`.

[L+96] J.C. Laprie *et al.* : *Guide De La Sûreté De Fonctionnement*. Cepadues, 1996.

[LA90] P.A. Lee et T. Anderson : *Fault Tolerance : Principles and Practice*. Springer-Verlag, Seraucus, NJ, USA, 1990. isbn : 0387820779.

[Lab03] D. Labbé : *Corneille Dans L'Ombre de Molière – Histoire d'une découverte*. Les Impressions Nouvelles, 2003. isbn : 2-906131-65-2.

[Lap04] J.C. Laprie : Sûreté de fonctionnement des systèmes : concepts de base et terminologie. *La Revue de l'Electricité et de l'Electronique*, num. 11, pp. 95–105, 2004.

[Lau02] A.P. Laudicina : Nessus – a powerful, free remote security scanner. *Sys Admin : The Journal for UNIX Systems Administrators*, vol. 11, num. 5, pp. 24, 26, 28–30, 2002.

[LE01] D. Larochelle et D. Evans : Statically detecting likely buffer overflow vulnerabilities. *Dans Proceedings of the 10th conference on USENIX Security Symposium (SSYM'01)*, pp. 14–14, Berkeley, CA, USA, 2001.

[LHF+00] R. Lippmann, J.W. Haines, D.J. Fried, J. Korba et K. Das : Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation. *Dans Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection (RAID'00)*, pp. 162–182, London, UK, 2000. Springer-Verlag.

[LIV] Living Internet. Site d'information concernant Internet, Adresse Internet : `www.livinginternet.com`.

[LL01] C. Labbé et D. Labbé : Inter-textual distance and authorship attribution Corneille and Molière. *Journal of Quantitative Linguistics*, vol. 8, num. 3, pp. 213–231, 2001.

[LMD05] C. Leita, K. Mermoud et M. Dacier : ScriptGen : an automated script generation tool for honeyd. *Dans 21st Annual Computer Security Applications Conference (ACSA'2005)*, pp. 203–214, Tucson, AZ, USA, décembre 2005.

[LW05] K.W. Lye et J.M. Wing : Game strategies in network security. *International Journal of Information Security*, vol. 4, num. 1, pp. 71–86, février 2005.

[MAF03] MAFTIA, Malicious and Accidental-Fault Tolerance in Internet Applications : conceptual model and architecture. MAFTIA Project Deliverable D21, 2003. Project IST-1999-11583.

[McH00] J. McHugh : The 1998 Lincoln Laboratory IDS Evaluation. *Dans Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection (RAID'00)*, pp. 145–161, London, UK, 2000. Springer-Verlag.

[MISRA] The Motor Industry Software Reliability Association : The misra Home Page. Adresse Internet : `www.misra.org.uk`.

[MM01] L. Mé et C. Michel : Intrusion detection : A bibliography. Rapport technique SSIR-2001-01, Supélec, Rennes, France, 2001.

[MMM+01] L. Mé, Z. Marrakchi, C. Michel, H. Debar et F. Cuppens : La détection d'intrusions : les outils doivent coopérer. *La Revue de l'Electricité et de l'Electronique*, num. 5, pp. 50–55, 2001.

[MPS+03] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford et N. Weaver : The spread of the sapphire/slammer worm. Adresse Internet : `www.caida.org/outreach/papers/2003/sapphire/sapphire.html`, 2003.

[MSB02] D. Moore, C. Shannon et J. Brown : Code-red : a case study on the spread and victims of an internet worm. *Dans Proceedings of the Internet Measurement Workshop (IMW)*, 2002.

[MSB+06] D. Moore, C. Shannon, D.J. Brown, G.M. Voelker et S. Savage : Inferring internet denial-of-service activity. *ACM Trans. Comput. Syst.*, vol. 24, num. 2, pp. 115–139, 2006, New-York, NY, USA.

[MSVS04] D. Moore, C. Shannon, G.M. Voelker et S. Savage : Network telescopes : Technical report. Rapport technique TR-2004-04, Cooperative Association for Internet Data Analysis - CAIDA, 2004.

[MW98] U. Maurer et S. Wolf : Diffie-Hellman, Decision Diffie-Hellman, and Discrete Logarithms. *Dans Proceedings of IEEE International Symposium on Information Theory (ISIT'98)*, pp. 327, 1998.

[NET] The GNU Netcat project. Adresse Internet : `netcat.sourceforge.net`.

[NST04] D.M. Nicol, W.H. Sanders et K.S. Trivedi : Model-based evaluation : From dependability to security. *IEEE Transactions on Dependable and Secure Computing*, vol. 01, num. 1, pp. 48–65, 2004, Los Alamitos, CA, USA.

[ODK99] R. Ortalo, Y. Deswarte et M. Kaaniche : Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security. *IEEE Transactions on Software Engineering*, vol. 25, num. 5, pp. 633–650, 1999, Los Alamitos, CA, USA.

[Ove07] M. Overton : The journey, so far : Trends, graphs and statistics. *Dans Proceedings of the Virus Bulletin 2007 Conference*, 2007. Vienne, Autriche.

[PDD03] F. Pouget, M. Dacier et H. Debar : White paper : honeypot, honeynet, honeytoken : terminological issues. Rapport technique EURECOM+1275, Institut Eurecom, Sophia-Antipolis, France, 2003.

[PDP05] F. Pouget, M. Dacier et V.H. Pham : Leurre.com : on the advantages of deploying a large scale distributed honeypot platform. *Dans E-Crime and Computer Conference (ECCE'05)*, Monaco, Monaco, 2005.

[PF94] V. Paxson et S. Floyd : Wide-area traffic : the failure of poisson modeling. *Dans Proceedings of the conference on Communications architectures, protocols and applications (SIGCOMM'94)*, pp. 257–268, London, UK, 1994. ACM Press. isbn : 0-89791-682-4.

[PH07] N. Provos et T. Holz : *Virtual Honeypots : From Botnet Tracking to Intrusion Detection*. Addison-Wesley Professional, 2007.

[Pir07] L. Piriou : Téléphonie mobile : un pirate démasqué. www.infos-du-net.com/actualite/10987-Mobiles-virus.html, 2007.

[Pla05] V. Planchon : Traitement des valeurs aberrantes : concepts actuels et tendances générales. *Biotechnologie, agronomie, société et environnement*, vol. 9, num. 1, pp. 19–34, 2005.

[Pou06] F. Pouget : *Distributed system of honeypot sensors : discrimination and correlative analysis of attack processes*. Thèse de doctorat, Institut Eurécom, Sophia-Antipolis, France, 2006.

[pro] Caida project : Caida analysis of code-red. Adresse Internet : www.caida.org/analysis/security/code-red.

[Pro03] The Honeynet Project : Know your enemy : Sebek, 2003. Adresse Internet : www.honeynet.org/papers/sebek.pdf.

[Pro04] N. Provos : A Virtual Honeypot Framework. *Dans Proceedings of the 13th conference on USENIX Security Symposium (SSYM'04)*, San Diego, CA, USA, 2004.

[Pro05] Honeynet Project : Know Your Enemy : Honeywall CDROM Roo. Adresse Internet : www.honeynet.org, 2005.

[PS98] C. Phillips et L. Painton Swiler : A graph-based system for network-vulnerability analysis. *Dans Proceedings of the 1998 workshop on New security paradigms (NSPW'98)*, pp. 71–79, Charlottesville, Virginia, USA, 1998. ACM Press. isbn : 1-58113-168-2.

[PZC+96] N. Puketza, K. Zhang, M. Chung, B. Mukherjee et R. Olsson : A Methodology for Testing Intrusion Detection Systems. *IEEE Transactions on Software Engineering*, vol. 22, num. 10, pp. 719–729, 1996, Piscataway, NJ, USA.

[R-D07] R-Development Core Team : *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienne, Autriche, 2007. isbn : 3-900051-07-0, Adresse Internet : www.R-project.org.

[RBC07] D. Ramsbrock, R. Berthier et M. Cukier : Profiling Attacker Behavior Following SSH Compromises. *Dans International Conference on Dependable Systems and Networks (DSN'07)*, pp. 119–124, Washington, DC, USA, 2007.

[RJMT06] M. Rajad, Z. Jay, F. Monrose et A. Terzis : A multifaceted approach to understanding the botnet phenomenon. *Dans Internet Measurement Conference*, pp. 41–52, 2006.

[rp05] ripe et pluf : Advanced Antiforensics : Self. *Phrack 63*, vol. 0x0b, num. 0x3f, 2005. Adresse Internet : www.phrack.org.

[Sav] A. Savochkin : OpenVZ. Adresse Internet : openvz.org.

[Sch99] B. Schneier : Attack trees. *Dr. Dobb's Journal*, 1999. Adresse Internet : www.schneier.com/paper-attacktrees-ddj-ft.html.

[SCK04] D.P. Siewiorek, R. Chillarege et Z. Kalbarczyk : Reflections on Industry Trends and Experimental Research in Dependability. *IEEE*

Trans. Dependable Secur. Comput., vol. 1, num. 2, pp. 109–127, 2004, Los Alamitos, CA, USA.

[Sei] C. Seifert : Malicious Ssh Login Attempts. Adresse Internet : `www.securityfocus.com/infocus/1876`.

[SHJ+02] O. Sheyner, J. Haines, S. Jha, R. Lippmann et J.M. Wing : Automated Generation and Analysis of Attack Graphs. *Dans Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pp. 273–284, Oakland, CA, USA, 2002. IEEE Computer Society.

[SHK06] K. Sallhammar, B.E. Helvik et S.J. Knapskog : A Game-Theoretic Approach to Stochastic Security and Dependability Evaluation. *Dans Proceedings of the 2nd IEEE International Symposium on Dependable, Automatic and Secure Computing (DASC'06)*, pp. 61–68, Indianapolis, IN, USA, 2006. IEEE Computer Society.

[SMQ93] W. Stute, W. Gonzales Manteiga et M. Presedo Quindimil : Bootstrap based goodness-of-fit-tests. *Metrika*, vol. 40, num. 1, pp. 243–256, 1993.

[Sno02] Snort : The open source network intrusion detection system. Adresse Internet : `www.snort.org`, 2002.

[SPE] Specter Intrusion Detection System. Adresse Internet : `www.specter.com`.

[Spi02] L. Spitzner : *Honeypots : Tracking Hackers*. Addison-Wesley Professional, 2002. isbn : 0321108957.

[Spi03a] L. Spitzner : The honeynet project : Trapping the hackers. *IEEE Security and Privacy*, vol. 1, num. 2, pp. 15–23, 2003.

[Spi03b] L. Spitzner : Honeytokens : The Other Honeypot, 2003. Adresse Internet : `www.securityfocus.com/infocus/1713`.

[SPW02] S. Staniford, V. Paxson et N. Weaver : How to Own the Internet in Your Spare Time. *Dans Proceedings of the 11th USENIX Security Symposium*, pp. 149–167, Berkeley, CA, USA, 2002.

[ste01] stealth : It cuts like a knife – SSHarp. *Dans Phrack 59*, volume 0x0b, 2001. Adresse Internet : `www.phrack.org`.

[Sto88] C. Stoll : Stalking the wily hacker. *Commun. ACM*, vol. 31, num. 5, pp. 484–497, 1988, New-York, NY, USA.

[SWT01] D. Song, D. Wagner et X. Tian : Timing analysis of keystrokes and timing attacks on SSH. *Dans Proceedings of the 10th conference on USENIX Security Symposium (SSYM'01)*, pp. 25–25, Berkeley, CA, USA, 2001.

[TCS85] DOD, U.S. Departement of Defense, " Trusted Computer Security Evaluation Criteria (TCSEC)". 5200.28-STD, 1985. Adresse Internet : `www.fas.org/irp/nsa/rainbow/std001.htm`.

[Tec92] National Institute of Standards and Technology : Federal Criteria for Information Technology Security. Draft, Volume I et II, National Institute of Standards and Technology (NIST) and National Security Agency (NSA), 1992.

[Uni] University of Cambridge Computer Laboratory : The Xen virtual machine monitor. Adresse Internet : www.cl.cam.ac.uk/research/srg/netos/xen.

[URLa] Kernel Based Virtual Machine. Adresse Internet : kvm.qumranet.com/kvmwiki.

[URLb] Linux Virtualization Wiki. Adresse Internet : virt.kernelnewbies.org.

[VAC+] A. Valdes, M. Almgren, S. Cheung, Y. Deswarte, B. Dutertre, J. Levy, H. Saidi, V. Stavridou et T.E. Uribe : An architecture for an adaptive intrusion tolerant server. *Dans Proceedings of the 10th International Workshop on Security Protocols*, Lecture Notes in Computer Science – LNCS, Cambridge, UK. Springer-Verlag. isbn : 978-3-540-20830-3, Adresse Internet : www.csl.sri.com/papers/protocols_sri_02.

[VH04] E. Vanderviere et M. Hubert : An adjusted boxplot for skewed distributions. *Dans Proceedings of the 16th Symposium of IASC (COMPSTAT'04)*, pp. 1933–1940, Prague, République tchèque, 2004.

[VMw] VMware Inc. : Vmware. Adresse Internet : www.vmware.com.

[VNC+06] P. Verissimo, N. F. Neves, C. Cachin, J. Poritz, D. Powell, Y. Deswarte, R. Stroud et I. Welch : Intrusion-Tolerant Middleware : The Road to Automatic Security. *IEEE Security and Privacy*, vol. 4, num. 4, pp. 54–62, 2006.

[Wil91] R. Williams : An extremely fast ziv-lempel data compression algorithm. *Dans Data Compression Conference*, pp. 362–371, 1991.

[WMT03] D. Wang, B.B. Madan et K.S. Trivedi : Security analysis of SITAR intrusion tolerance system. *Dans Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems (SSRS'03)*, pp. 23–32, Fairfax, VA, USA, 2003. ACM Press.

[WTF99] H. Waeselynck et P. Thévenod-Fosse : A Case Study in Statistical Testing of Reusable Concurrent Objects. *Dans Proceedings of the Third European Dependable Computing Conference on Dependable Computing*, pp. 401–418, London, UK, 1999. Springer-Verlag.

[XJ04] D. Xu X. Jiang : Collapsar : a VM-based architecture for network attack detention center. *Dans Proceedings of the 13th conference on USENIX Security Symposium (SSYM'04)*, San Diego, CA, USA, 2004.

[YL06] T. Ylonen et C. Lonvick : The Secure Shell (Ssh) Authentication Protocol, 2006. Adresse Internet : www.ietf.org/rfc/rfc4252.txt.

[ZC06] C.C. Zou et R. Cunningham : Honeypot-aware advanced botnet construction and maintenance. *Dans International Conference on Dependable Systems and Networks (DSN'06)*, pp. 199–208, Philadelphie, PA, USA, 2006.

[ZGT02] C. Zou, W. Gong et D. Towsley : Code Red Worm Propagation Modeling and Analysis. *Dans Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 138–147, novembre 2002. Adresse Internet : citeseer.ist.psu.edu/zou02code.html.

[ZGTG05] C.C. Zou, W. Gong, D. Towsley et L. Gao : The monitoring and early detection of internet worms. *IEEE/ACM Trans. Netw.*, vol. 13, num. 5, pp. 961–974, 2005, Piscataway, NJ, USA.

Observation, caractérisation et modélisation de processus d'attaque sur Internet

Le développement de méthodes permettant l'observation et la caractérisation d'attaques sur Internet est important pour améliorer notre connaissance sur le comportement des attaquants. En particulier, les informations issues de ces analyses sont utiles pour établir des hypothèses réalistes et mettre en oeuvre des mécanismes de protection pour y faire face. Les travaux présentés dans cette thèse s'inscrivent dans cette optique en utilisant des pots de miel comme moyen pour collecter des données caractérisant des activités malveillantes sur Internet. Les pots de miel sont des systèmes informatiques volontairement vulnérables et visant à attirer les attaquants afin d'étudier leur comportement.

Nos travaux et contributions portent sur deux volets complémentaires. Le premier concerne le développement d'une méthodologie et de modèles stochastiques permettant de caractériser la distribution des intervalles de temps entre attaques, la propagation et les corrélations entre les processus d'attaques observés sur plusieurs environnements, en utilisant comme support les données issues de pots de miel basse interaction déployés dans le cadre du projet Leurré.com. Le deuxième volet de nos travaux porte sur le développement et le déploiement d'un pot de miel haute interaction permettant d'étudier aussi la progression d'une attaque au sein d'un système, en considérant comme exemple des attaques visant le service ssh. L'analyse des données collectées nous a permis d'observer différentes étapes du processus d'intrusion et de montrer la pertinence de notre d'approche.

Mots-clefs : Sécurité, Internet, Pots de miel, Evaluation quantitative, Intrusion

Observation, characterization and modeling of attack processes on the Internet

The development of appropriate methods to observe and characterize attacks on the Internet is important to improve our knowledge about these threats and the behavior of the attackers. In particular, information obtained from such analyses are useful to establish realistic assumptions and to implement protection mechanisms to cope with these threats. The work presented in this thesis falls within this context using honeypots as a means to collect data characterizing the malicious activities on the Internet. A honeypot is a computer system that is deliberately vulnerable and is aimed at attracting the attackers to study their behavior.

Our work and contributions cover two main objectives. The first one concerns the development of a methodology and stochastic models to characterize the distribution of the time intervals between attacks, the propagation of attacks and the correlations between the attack processes observed on several honeypot environments, using data collected from low interaction honeypots deployed in the context of the Leurré.com project. The second part of our work focuses on the development and deployment of a high interaction honeypot to explore the progression of an attack within a system, considering as an example attacks against the ssh service. The analysis of data collected allowed us to observe different stages of an intrusion and to demonstrate the relevance of our approach.

keywords : Security, Internet, Honeypot, Quantitative evaluation, Intrusion