Un WOMBAT pour évaluer la cybercriminalité

C. Leita, V.H. Pham, O. Thonnard, M. Dacier et E. Kirda

Symantec Research Labs Europe, Sophia Antipolis
{corrado_leita,marc_dacier}(@)symantec.com
EURECOM, Sophia Antipolis
{pham,thonnard,kirda}(@)eurecom.fr

Résumé Le projet WOMBAT est un projet de recherche collaboratif financé par la Commission Européenne. Il vise à mieux comprendre les menaces existantes et émergeantes qui visent l'économie de l'Internet. L'approche menée par les partenaires inclut un effort de collecte de données ainsi que le développement de nouveaux outils d'analyse. Dans ce document, nous présentons l'une des sources d'information utilisées dans le projet ainsi que les premiers résultats obtenus. Nous invitons le lecteur intéressé par une description plus complète du projet WOMBAT en lui même à consulter les actes du premier workshop WOMBAT [21] où de plus amples informations sur la globalité du projet sont disponibles ainsi que sur d'autres initiatives complémentaires en cours dans le monde.

Avant-Propos Dans [21], les auteurs offrent une présentation très détaillée d'un outil de collecte de données particulier utilisé dans le cadre du projet WOMBAT. La présentation détaille les schémas entité relations utilisés dans la base de données. Dans ce qui suit, nous désirons offrir au lecteur une première synthèse des résultats obtenus grâce à l'analyse des données collectées à l'aide de cette infrastructure. Cependant, afin d'améliorer la lisibilité de ce document, nous commençons par rappeler les motivations justifiant ce travail ainsi que les grandes lignes de l'infrastructure de collecte de données. Le lecteur qui serait déjà familier des travaux publiés dans [21] peut s'abstenir de lire cette section et passer directement à la présentation des résultats.

1 Introduction

Afin de mieux protéger l'économie de l'Internet, nos systèmes d'information et les utilisateurs de l'Internet, il est primordial de bien comprendre les menaces existantes et émergeantes. Dans ce but, il faut être capable de réaliser des mesures sur les processus d'attaques en cours dans le monde entier. Ce type d'études expérimentales reçoit une attention croissante depuis quelques années et plusieurs initiatives très intéressantes existent qui nous permettent d'observer des activités malveillantes ou de capturer des copies de codes binaires malveillants (que nous appellerons malware par la suite). Des contributions importantes ont été faites notamment dans les domaines

tels que : i) les télescopes de l'Internet et l'étude de ce que l'on appelle les Darknets [23,35,30], ii) l'utilisation de pots de miel de basse ou haute interaction [13,34,31,2,41], et iii) d'autres initiatives visant à rassembler et partager des journaux de garde barrières et autres outils de détections d'intrusions [14].

Le projet Leurré.com a démarré en 2003 et a, depuis, été intégré et développé dans le cadre du projet WOMBAT. Il repose sur un système distribué de pots de miel situés dans plus de 30 pays différents sur les 5 continents. L'objectif principal de cette infrastructure est d'avoir une vision plus objective de certaines classes d'attaques observables sur l'Internet et, ce, à l'aide de données quantitatives non biaisées obtenues sur une longue période de temps. Nous avons décidé de conserver dans une base de données centralisée un ensemble très riche d'informations en relation avec un nombre limité de machines que nous surveillons de près. De façon très concrète, cela signifie que nous avons déployé des pots de miel utilisant la technologie Honeyd [31] aux bords des réseaux de partenaires situés un peu partout dans le monde. Dans le cadre de WOMBAAT, nous avons enrichi cette plateforme de collecte en construisante et en déployant une nouvelle génération de pots de miel utilisant cette fois la technologie Scriptgen [20,18,17]. Ces nouvelles sondes améliorent grandement l'interaction offerte avec les attaquants et enrichissent ainsi les données collectées. Nous enregistrons tous les paquets envoyés de et vers ces machines et nous les stockons dans une base de données en les enrichissant de nombreuses informations contextuelles et de méta-données caractérisant les sessions d'attaque. Dans les sections qui suivent, nous présentons ces deux infrastructures de collecte de données et, ensuite, offrons une synthèse des résultats d'analyse obtenus par les partenaires du projet WOMBAT à l'aide des données mises à leur disposition.

La structure de ce document est la suivante : la section 2 présente l'infrastrucutre de collecte de données initiale basée sur le déploiement de pots de miel basse interaction. Nous donnons quelques exemples révélateurs du type d'information qu'ils offrent. La section 3 présente comment le système nommé SGNET enrichit cette première plateforme. Ce système est, comme le premier, utilisable gratuitement par tout qui accepte de déployer une de nos sondes.

La Section 4 présente la notion de ce que nous appelons des *évènements d'attaque*. Il s'agit de périodes de temps relativement courtes durant lesquelles des observations particulières sont constatées. L'identification de tels évènements permet de voir ce que nous supposons être des *armées de zombies*, certaines restant apparemment visibles durant plus de 700 jours ! La section 5 va plus loin dans l'analyse de ces traces, mettant en lumière l'utilité d'appliquer une analyse multi-critères à ces évènements. La Section 6 livre quelques idées sur le type d'information contextuelle que SGNET offre sur les malwares qu'il collecte. Des exemples concrets sont donnés qui démontrent l'utilité

de ces informations dans notre quête de nouvelles menaces et pour une meilleure compréhension des liens entre la phase d'injection de code, le code shell injecté et le malware en tant que tel. Enfin, la Section 7 conclut ce document.

2 Leurre.com v1.0 Honeyd

2.1 Rappel historique

L'institut EURECOM a commencé à collecter des traces d'attaques sur l'Internet en 2003 à l'aide de pots de miel. La toute première plateforme était composée de trois pots de miel dits de haute interaction construits à l'aide de la technologie de virtualisation VMWARE (nous invitons le lecteur intéressé par les détails de cette configuration à se reporter à [12]). Il a été montré dans [12,11] que ces premières expérimentations ont permis de mettre en avant le principe de localisation des attaques sur l'Internet : les attaques observées sont différentes selon le point de collecte. Pour confirmer ces premiers résultats, plusieurs pots de miel identiques ont été déployés à plusieurs endroits différents. Malheureusement, la solution VMWARE ne permettait pas de passer facilement à l'échelle. Tout d'abord, elle était coûteuse en termes de maintenance. Ensuite, elle nécessitait des ressources importantes (CPU, mémoire). Enfin, afin d'éviter tout problème légal, il fallait s'assurer que ces machines ne pourraient être compromises et utilisées à des fins malveillantes, chose très difficile à assurer de façon certaine ave ce type de système. Pour toutes ces raisons, le choix s'est porté vers une solution de type basse interaction basée sur honeyd [31]. Ceci permettait de déployer des plateformes à bas coût, faciles à maintenir et avec un risque de sécurité très faibles, autant d'ingrédients nécessaires pour demander à des partenaires d'accepter d'héberger de telles sondes sur la base du volontariat. Cette approche permit de construire un réseau de près de 50 sondes distribuées dans le monde entier dont les données collectées étaient stockées dans une base de données relationelle centralisée accessible à tous les partenaires. L'identité des participants à cette initiative ainsi que celle des attaquants est protégée par un accord de confidentialité signé par chaque participant.

2.2 Définitions

Nous décrivons dans ce qui suit quelques aspects techniques importants en rapport avec l'architecture de la plateforme, les mécanismes de collecte des journaux, le chargement de la base de données et le mécanisme d'enrichissement des données.

Architecture de la plateforme : Comme nous l'avons dit, l'objectif principal est de comparer du trafic réseau non sollicité, capturé en divers endroits. Pour que ces comparaisons aient du sens, l'architecture doit être identique en tous points. Nous avons configuré Honeyd pour simuler 3 hôtes virtuels utilisant trois adresses IP consécutives. Nous avons configuré le système de personnalité de Honeyd pour émuler deux types de système différents : deux machines prétendent être des machines sous Windows 2000 SP3 et la troisième un Linux Kernel 2.4.20. Les machines Windows ont les ports suivants ouverts : FTP, Telnet, Serveur WEB, Netbios name service, Netbios session service et Service Message Block. La machine Linux, elle, offre les services suivants : serveurs FTP, SSH et Web, Proxy sur les ports 8080 et 8081, remote shell, LPD Printer service et portmapper.

Une quatrième adresse IP est utilisée afin d'avoir accès à la machine physique où tourne Honeyd et réaliser les tâches de maintenance à distance. Nous utilisons tcpdump [36] pour capturer l'entièreté des traces réseau sur chaque plateforme. Par sécurité, un garde barrière inversé protège le système : il accepte les connexions entrantes mais interdit toute connexion sortante. L'accès à la machine hôte n'est possible que via une connexion ssh utilisable dans une fenêtre de deux heures par jour et réalisable uniquement depuis nos serveurs dédiés à la maintenance.

Mécanisme de collection de données : Chaque jour, à l'aide d'une connexion chiffrée, nous rapatrions les fichiers tcpdump. Tous les fichiers de trace sont stockés sur un serveur central.

Mécanisme de chargement des données : L'étape suivante consiste à charger ces fichiers dans une base de données Oracle. Les fichiers tcpdump sont traités par un certain nombre de scripts qui transforment les données brutes en un certain nombre de concepts de plus haut niveau, tout en stockant également les données initiales dans la base. Nous détaillons ci dessous quelques uns des concepts utilisés dans la base.

1. Source : Une Source correspond à une adresse IP qui a envoyé au moins un paquet à au moins une de nos plateformes. Il est important de comprendre qu'à une adresse IP peut correspondre plusieurs Sources distinctes. En effet, une adresse IP n'est liée à une Source donnée que pour autant que moins de 25 heures s'écoulent entre deux paquets reçus. Une fois passé ce délai, si cette adresse IP recontacte une de nos plateformes, elle sera associée à une nouvelle Source. En groupant les traces par Source et non par adresse IP, nous minimisons le risque d'associer des traces émises par des machines différentes qui auraient reçu, au cours du temps, la même adresse IP.
2. Grande_Session : C'est l'ensemble des paquets échangés entre une Source et une sonde. Une Grande Session est caractérisée par la durée de l'attaque, le nombre

de paquets envoyés par la Source, le nombre de machines virtuelles sur la sonde contactées par la Source, ...

3. Séquence de Ports : Une Séquence de Ports est la séquence ordonnée dans le temps des ports (sans redondance) auxquels une source a envoyé des paquets à une machine virtuelle. Par exemple, si un attaquant envoie les paquets suivants vers une machine virtuelle : ICMP, 135 TCP, 135 TCP, 139 TCP, la Séquence de Ports associée sera notée par la chaîne $I|135T|139T$. Toute Grande Session peut avoir, au plus, trois Séquences de Portes qui lui sont associées (une par machine virtuelle sur la sonde).

   Cette notion est importante car elle nous permet par la suite de classer les attaques en différents groupes. En effet, comme cela est expliqué en détail dans [12], la plupart des outils d'attaque sont automatisés et déterministes : un outil donné génère donc toujours la même Séquence de Ports lorsqu'il parle à une victime potentielle.

4. Cluster : Un Cluster regroupe l'ensemble des Sources qui ont laissé la même empreinte sur une sonde. Nous utilisons un algorithme de clustering sur le trafic généré par les Sources. La première étape de cette algorithme consiste à regrouper les Grande_Sessions dans des *sacs*. Cette étape vise à différencier certaines classes d'activités en prenant en considérant un ensemble de discriminants simples tels que le nombre d'hôtes virtuels contactés et la liste des Séquences de Ports. Ces *sacs* sont ensuite raffinés à l'aide d'autres paramètres des Grande_Sessions : leur durée, nombre total de paquets échangés, intervalle de temps moyen entre paquets, nombre de paquets vers chaque machine virtuelle. Ces paramètres peuvent prendre n'importe quelle valeur dans l'intervalle $[0, \infty]$ et nous segmentons cet intervalle afin de créer des classes de valeurs. Ceci est fait à l'aide d'un algorithme dit de *peak picking*. Les Grande_Sessions appartenant à un même *sac* et partageant les mêmes paramètres détaillés sont regroupés au sein de Clusters. La dernière étape concerne la validation du payload. L'algorithme étudie la concaténation de tous les payloads envoyés par un attaquant au sein d'une Grande_Session. Si nous détectons au sein d'un cluster plusieurs groupes distincts de Grande_Sessions qui partagent des payloads communs, nous scindons ce cluster en plusieurs clusters distincts. En résumé, par construction, un Cluster regroupe un ensemble de Grande_Sessions qui, vraisemblablement, ont pour cause commune un même outil d'attaque.

5. Une série temporelle de Cluster $\Phi_{T,c}$ est une fonction définie sur une période de temps T, T définie comme étant un intervalle de temps exprimé en jours. Cette fonction renvoie le nombre de Sources associées au Cluster c par jour au cours de l'intervalle défini.

6. L'observation d'une série temporelle de Cluster $\Phi_{T,c,op}$ est une fonction définie sur une période de temps T, T définie comme étant un intervalle de temps exprimé en jours. Cette fonction renvoie le nombre de Sources associées au Cluster c observable à partir d'un *point d'observation particulier op*. Ce point d'observation peut être une sonde particulière ou un pays d'origine particulier des attaquants. Dans le premier cas, $\Phi_{T,c,platform_X}$ renvoie, par jour, le nombre de Sources appartenant au Cluster c qui ont visé la plateforme $platform_X$. De même, dans le second cas, $\Phi_{T,c,country_X}$ renvoie, par jour, le nombre de sources appartenant au cluster c qui appartiennent au pays $country_X$. Il est clair que nous aurons toujours : $\Phi_{T,c} = \sum^{\forall i \in countries} \Phi_{T,c,i} = \sum^{\forall x \in platforms} \Phi_{T,c,x}$

Enrichissement de l'information : Pour enrichir l'information liée à chaque Source, nous lui adjoignons trois types d'information :

1. **Information géographique** : Pour obtenir l'information liée à la localisation géographique d'une adresse IP telle que : l'organisation détentrice, le fournisseur de service Internet (ISP) et le pays d'origine nous avons tout d'abord utilisé le service Netgeo [25], développé dans le cadre du projet CAIDA. A notre grande surprise, les Pays-Bas et l'Australie apparaissaient comme les pays les plus actifs en termes d'attaques. A des fins de vérification, nous avons utilisé un autre service, Maxmind [22]. Ceci nous a mené à identifier un certain nombre de problèmes avec les résultats fournis par Netgeo. [29] compare les résultats offerts par ces deux outils. Il en ressort que les Pays Bas et l'Australie ne doivent pas être considérés comme les pays les plus agressifs en termes d'attaques sur Internet.
2. **OS Fingerprint** : Afin de déterminer le système d'exploitation des attaquants, nous utilisons les techniques dites d' *OS fingerprinting*. Nous utilisons les outils disco [1] et p0f [42]. Il ressort de notre expérience que p0f est plus précis que disco. Nous avons renoncé à utiliser les techniques actives de fingerprinting, telles que Nmap, Quezo, ou Xprobe afin d'éviter d'alerter les attaquants de nos investigations.
3. **Nom de domaine :** Nous réalisons également en temps réel une recherche inverse de nom de domaine (DNS reverse name lookup) afin d'obtenir, lorsque c'est possible, le nom associé à une machine envoyant des paquets à l'une de nos sondes.

2.3 Aperçu général

La Figure 1 (gauche) représente l'évolution du nombre de plateformes. Chaque courbe correspond à la durée de vie d'une plateforme. Comme on peut le voir, la collecte de données a commencé en Janvier 2003 avec un pot de miel VMWARE et nous avons commencé à déployer les pots de miel basse interaction en Avril 2004.

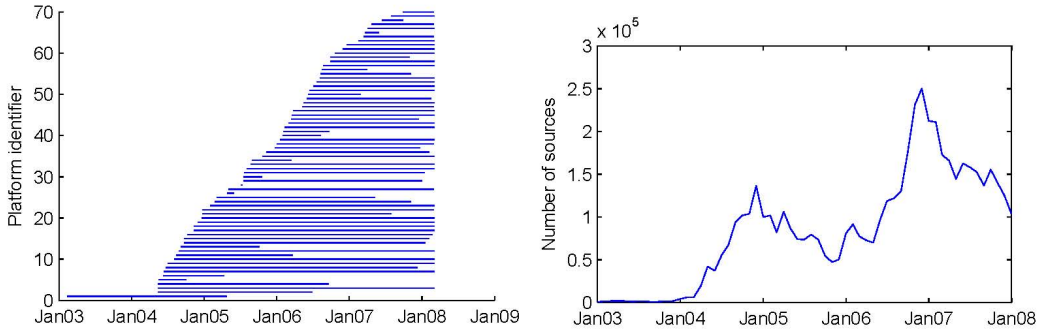


Fig. 1. Gauche : évolution du nombre de plateformes, Droite : nombre de Sources

Depuis, le nombre de partenaires rejoignant le projet n'a fait que croître. Au total, nous avons environ 50 partenaires actifs et 20 inactifs. Ces sondes ont, au total, couvert 37 réseaux /8 différents, situés dans 28 pays différents situés sur les 5 continents. Au total, nous avons observé 5173624 Sources correspondant à 3461745 adresses IP différentes. La Figure 1 (droite) représente l'évolution du nombre de Sources au cours du temps. La forme de la courbe est, bien entendu, influencée par le nombre de plateformes actives. Jusqu'en Avril 2004, le trafic reste très faible. Après cela, le nombre de Sources augmente. Il est intéressant de constater que le nombre de Sources au cours des six derniers mois de 2004 est bien plus important qu'au cours des six derniers mois de 2005 bien que nous ayons eu plus de plateformes actives à ce moment là. Au total, nous avons observé 155041 Clusters différents.

La Figure 2 (gauche) représente la fonction de distribution cumulée du nombre de Sources par Cluster. Le point (X,Y) sur la courbe signifie que Y*100% du nombre total de Clusters contient moins que X sources. Comme on peut le voir, la plupart des clusters sont donc très petits. Il y a, en fait, seulement 15521 Clusters contenant plus de 10 sources chacun. En interrogeant la base, on se rend compte que ces seuls clusters, totalisant près de 10% du nombre total de clusters, contiennent à eux seuls près de 95% du nombre de Sources. En d'autres termes, le gros des attaques se trouve dans un nombre limité de Clusters et il existe un très grand nombre d'activités réalisées par un tout petit nombre de sources. En termes de systèmes d'exploitation des attaquants, l'immense majorité, selon p0f, sont des machines Windows. Ceci confirme les premiers résultats publiés dans [12,11]. La Figure 3 représente les dix pays ayant fourni le plus grand nombre de Sources. Si ces statistiques sont sans surprise pour les deux premiers, USA et Chine, le Canada en troisième position est plus surprenant ainsi que

CS (anciennement Serbie et Monténégro) en cinquième position. Pour ce pays, une (et une seule) de nos plateformes est particulièrement visée par des attaques venant de ce pays. Elle est tellement attaquée que ce pays arrive très haut dans le classement bien qu'il soit pratiquement invisible sur les autres sondes ! Enfin, afin de montrer la diversité des attaques observées sur les différentes plateformes, la Figure 2 (droite) représente la distribution du nombre de clusters par plateforme. Chaque colonne représente le nombre de clusters distincts observés sur une sonde. Nous avons autant de colonnes que de sondes. Comme on peut le voir, les attaques sont très diverses. Sur certaines plateformes, nous n'observons qu'un faible nombre de clusters alors que d'autres sont visées par une grande diversité d'attaques.

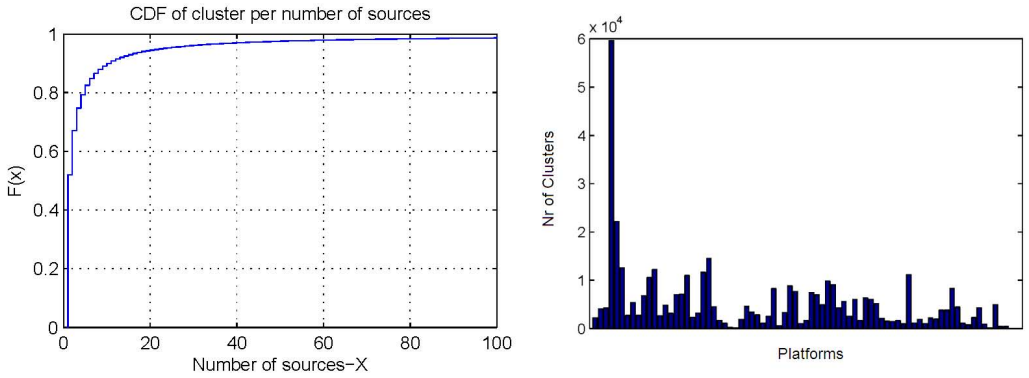


Fig. 2. Gauche : Fonction de distribution cumulée du nombre de sources par cluster ; Droite : Distribution du nombre de clusters par plateforme

2.4 Exemples Illustratifs

La grande diversité des données issues du monde réel, telles que les données des pots de miel, rend la tâche d'un analyste particulièrement difficile lorsqu'il s'agit de sélectionner et définir les caractéristiques importantes d'une attaque afin de parvenir à tirer des conclusions significatives sur les acteurs à la source de ces mêmes attaques. Pour illustrer ceci, nous offrons un ensemble d'exemples simples de la façon d'analyser les différentes facettes des menaces observées sur le réseau. Le but de cet exercice est de montrer que : i) plusieurs aspects différents d'une attaque peuvent offrir des éléments importants participant à la compréhension de la source de l'attaque, ii) certains processus d'attaques se manifestent au travers de ce que nous appelons des *évènements d'attaque* identifiables sur les plateformes. Ces évènements constituent

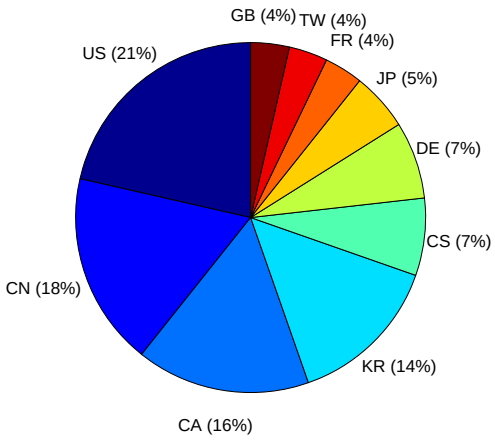


Fig. 3. Top 10 des pays attaquants

une des briques de base que nous pouvons utiliser pour construire une meilleure compréhension de l'écosystème des attaques.

Évolution au cours du temps des clusters L'analyse des séries temporelles offrent des informations très intéressantes (ex. tendances, changements abrupts, phénomènes émergeants) aux professionels de la sécurité en charge de la détection de phénomènes anormaux ou d'intrusions dans le trafic observé. La première illustration en est donnée par la Figure 4 qui montre l'évolution temporelle d'un cluster (ID 17718) du 1er Décembre 2006 au 1er Mars 2007. Le graphe de gauche montre une courbe regroupant les sources associées à ce cluster pour toutes les sondes prises ensemble. Le graphe de droite montre uniquement les sources observées sur les plateformes 14 d'une part et 42 d'autre part. Le même phénomène est clairement visible sur les deux sondes mais avec des amplitudes différentes. Il n'est cependant pas visible sur l'ensemble des sondes.

Localisation géographique des attaquants Poursuivant l'exemple précédent, il est intéressant de se pencher sur les pays d'origine des Sources associées au cluster 17718 durant l'évènement d'attaque identifié. Cette origine pourrait être un élément permettant de reconnaître le mode opératoire d'un groupe particulier.

Le résultat de la distribution géographique des sources liées au cluster 17718 est donnée dans la Table 1 : la première colonne donne le nom du pays (à l'aide de son

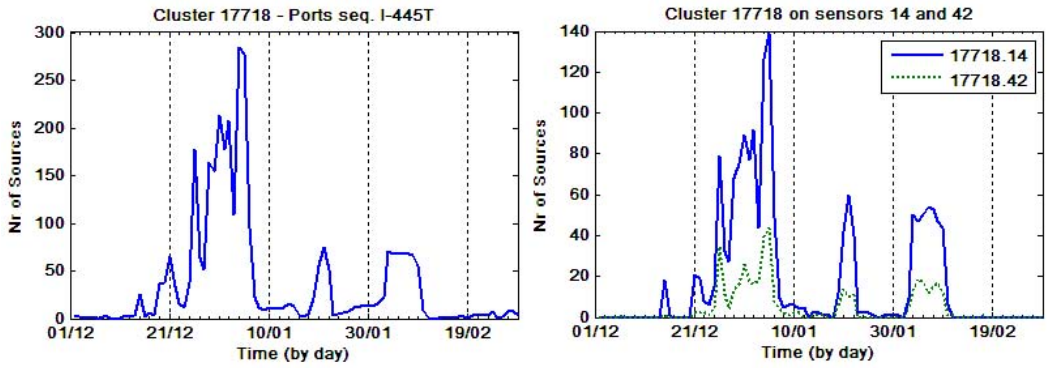


Fig. 4. Gauche : Évolution temporelle journalière globale du cluster 17718. Droite : Évolution temporelle journalière du même cluster pour les seules plateformes 14 et 42.

Tab. 1. Distribution géographique pour le cluster 17718 entre le 1er Déc. 06 et le 1er Mars 07

Pays d'origine	Nr de Sources	Pourcentage %
CN	1150	35.3
US	378	11.6
CA	255	7.8
FR	236	7.2
inconnu	215	6.6
TW	137	4.2
JP	128	3.9
IT	120	3.6
DE	107	3.3
Autres	524	16.1

code ISO), la deuxième donne le nombre de sources liées au cluster appartenant à ce pays, la troisième colonne indique le poids relatif de ce pays, en termes de sources attaquantes. Nous verrons par la suite (Section 5) comment ce type d'information agrégée peut être utilisé dans un cadre plus général de raisonnement sur les processus d'attaques.

Information sur les sous réseaux d'attaquants L'information sur le bloc d'adresses IP d'origine des attaquants complète assez bien la géolocalisation expliquée ci dessus. Au lieu de révéler des stratégies de placement liées à des considérations nationales, elles peuvent mettre en exergue des phénomènes liées à l'administration de certains blocs d'adresses par leurs propriétaires ainsi que sur les modes de propagation de certains malware. En effet, les blocs d'adresse IP des attaquants peuvent fournir une assez bonne indication de l'état de *propreté* de certains sous réseaux. Les réseaux les mieux gérés, les plus surveillés, se révéleront contenir moins d'attaquants que ceux où aucun soin particulier n'est porté à la sécurité. Ceci amène à trouver dans certains sous réseaux des concentrations de machine zombies participant à des botnets [8]. Ceci est renforcé par la tendance, montrée dans des études antérieures [7], de certains vers à se propager de façon préférentielle au sein des sous réseaux où ils ont réussi à entrer plutôt que de façon aléatoire sur l'Internet.

Les résultats de cette analyse sont donnés dans la Table 2[1] dans le cas d'une agrégation au niveau des /8. Une analyse semblable pourrait, bien entendu, être étendue aux blocs /16 our /24. Ce type d'information agrégée peut également être utilisée pour identifier, corréler des processus d'attaques apparemment différents mais qui partagent certaines caractéristiques spécifiques, telles que les sous réseaux utilisés préférentiellement pour se propager.

Plateformes ou sous réseaux ciblés Il a été rapporté que certains outils malveillants étaient à présent capables de générer des malwares adaptés à différentes régions géographiques [5]. Ceci permet aux cybercriminels de lancer des campagnes d'attaque mieux ciblées, propageant certains malwares aux seules régions géographiques ou sous réseaux qui les intéressent. Afin de valider ce type de scénario, nous pouvons analyser la relation qui existe entre les clusters et les plateformes où ils ont été observés. La Table 3 offre ce genre d'information. La première colonne donne l'identifiant d'une sonde et chaque ligne de la deuxième colonne indique le nombre de sources associées au cluster 17718 qui ont visé la sonde correspondante. La dernière colonne indique

[1] Afin de garantir la confidentialité liée aux adresses IP des attaquants, le premier byte des adresses a été modifié. Il ne s'agit donc pas des véritables préfixes mais nous avons gardé la relation de proximité entre préfixes qui existait dans les données initiales.

Tab. 2. Distribution anonymisée des /8 pour le cluster 17718, période du 1er Déc. 2006 au 1er Mars 2007.

Sous réseau d'origine - /8	Nr de sources
220.x.x.x	451
56.x.x.x	193
80.x.x.x	168
22.x.x.x	160
217.x.x.x	159
86.x.x.x	123
218.x.x.x	113
69.x.x.x	100
66.x.x.x	91
216.x.x.x	90
Autres	1602

le /8[2] dans lequel la sonde est située. Cette table offre une idée de plus du type de *point d'observation* que nous pourrons utiliser plus tard pour corréler les processus d'attaques.

Tab. 3. Distribution des plateformes visées par le cluster 17718

Plateforme visée	Nr de sources	/8
14	1552	139
76	871	134
42	431	150
57	70	24
71	67	58
53	42	88
55	42	83
Autres	175	-

3 Leurre.com v2.0 : SGNET

3.1 Comment augmenter le niveau d'interaction

Nous avons vu dans les sections précédentes comment nous avons pu générer des ensembles de données intéressants, nous offrant des informations quantitatives sur la localisation et sur l'évolution du trafic Internet malveillant. Nous avons été capables d'observer des comportements particuliers, souvent difficiles à expliquer ou à attribuer

[2] Ici aussi, les adresses des sondes ont été changées

à une seule origine. Il est, en effet, assez difficile de relier une observation particulière à une classe d'activités, et notre quête d'explications se trouvait confrontée à un manque de richesse d'information sur l'intention finale des assaillants. En effet, avec les pots de miel à basse interaction, nous ne pouvons pas continuer à interagir avec les Sources qui nous visent dans la mesure où nous ne *comprenons* pas les protocoles qu'elles nous parlent.

Ainsi, lors de notre expérience avec le projet Leurré.com, en raison de l'absence de scripts émulant la présence de réels services nous n'avons été capables que d'observer les premières requêtes de plusieurs phénomènes intéressants tels le déploiement du ver Blaster [6]. Comme ce ver n'envoie le code malveillant que lors d'un second échange de messages à la porte 135, nous n'avons pas vu le payload en lui même. C'est pourquoi il est difficile de distinguer dans nos traces, le trafic dû à Blaster de celui lié à toutes les autres attaques et scans qui visent la porte 135.

Heureusement, l'expérience a également montré que, même avec un tel niveau limité d'informations, un grand nombre d'analyses restait applicable et livrait des résultats nouveaux, intéressants. Pour pouvoir pallier à ce déficit d'interaction, nous devons trouver un moyen de laisser nos pots de miel parler avec l'assaillant. Cependant, dans la mesure où nous voulons continuer à déployer des sondes sur la base d'un partenariat volontaire, elles ne doivent ni ouvrir la voie à une attaque réelle ni être trop coûteuse en ressources. Cette double contrainte a mené au développement d'une approche nouvelle basée sur une technologie appelée *Scriptgen*.

3.2 ScriptGen

La technologie nommée *Scriptgen* [20,19] a été créée afin de pouvoir disposer de pots de miel avec le plus haut niveau possible d'interaction tout en utilisant très peu de ressources. Cela a été possible grâce à un mécanisme *d'apprentissage* du comportement d'un protocole réseau donné lorsqu'il est utilisé à des fins malveillantes par un outil d'attaque déterministe. Ce comportement appris est représenté sous la forme d'une machine à états finis qui est une implémentation partielle et incomplète du langage du protocole. Cette machine à états peut ensuite être utilisée pour répondre aux clients, reproduisant les réponses qu'auraient fournies un vrai serveur tout en ne nécessitant que très peu de ressources.

La phase d'apprentissage de Scriptgen est totalement agnostique du point de vue du protocole à découvrir : aucune connaissance préalable de celui ci n'est nécessaire, ni sur sa structure ni sur sa sémantique. Scriptgen est, en théorie, capable de rejouer toute conversation d'un protocole pour autant que le payload ne soit pas chiffré. Le module d'apprentissage de Scriptgen utilise en entrée un ensemble de traces réseau enregistrées lors d'interactions entre un attaquant et une véritable implémentation

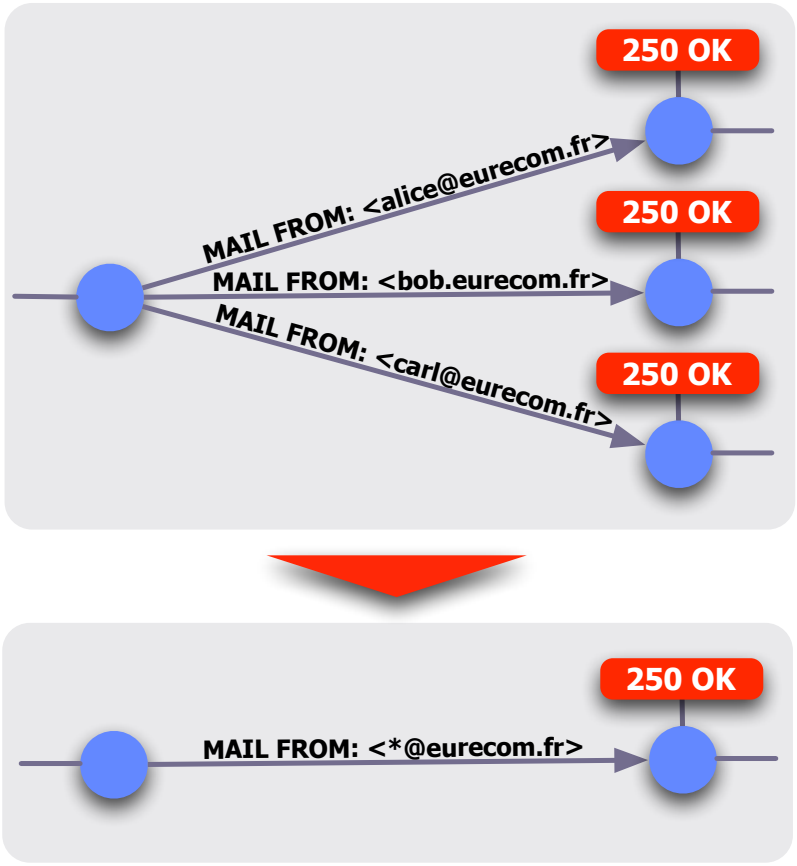


Fig. 5. Généralisation d'une machine à états finis ScriptGen

du serveur à émuler. Le coeur de la phase d'apprentissage est ce que nous appelons l'algorithme d'analyse de régions, introduit dans [20] il est basé sur des algorithmes d'alignement classiques issus de la bioinformatique [24]. Notre algorithme tire parti de la variabilité statistique des échantillons d'attaque observés afin d'identifier dans le flux de données les portions qui sont susceptibles d'avoir une grande importance sémantique. Ce critère de diversité dans les traces collectées est un élément très important à ne pas oublier lorsqu'on utilise Scriptgen. La Figure 5 montre un exemple d'abstraction sémantique pour un exemple hypothétique, et simplistique, de sous arbre d'une machine à états Scriptgen.

Les propriétés de l'approche Scriptgen sont telles qu'elles nous permettent de mettre en œuvre une approache d'apprentissage incrémentale totalement automatisée comme nous l'avons expliqué dans [19]. Les pots de miel utilisant la technologie ScriptGen sont capables de détecter lorsqu'une requête ne peut être satisfaite à l'aide de la seule connaissance incluse dans leur machine à états. Das ce cas, le pot de miel ne peut répondre à l'attaquant. Nous avons expliqué dans [19] comment le pot de miel peut réagir dans de telles circonstances en se référant temporairement à un véritable pot de miel à haute interaction qui joue alors le rôle d'un oracle. Le pot de miel, durant cette phase, se comporte comme un simple proxy entre l'attaquant et le pot de miel à haute interaction. Ceci permet au pot de miel de continuer à observer la conversation avec l'attaquant. Ce nouvel échange sera ensuite utilisé dans l'algorithme d'apprentissage pour raffiner la connaissance du pot de miel.

Scriptgen est capable d'apprendre correctement et d'émuler la conversation avec un assaillant pour des protocoles aussi complexes que NetBIOS [19]. ScriptGen permet donc de construire l'équivalent de pots de miel à haute interaction mais à un coût sans commune mesure. L'oracle nécessaire de temps à autre pour permettre l'observation de nouveaux échanges peut être hébergé dans un seul et même endroit et contacté par le biais d'un tunnel sécurisé, selon un concept similaire à celui présenté par Spitzner sous le nom de *honeyfarm*. Cependant, à la grande différence de cette approche, Scriptgen n'a besoin de l'oracle qu'un nombre très limité de fois, seulement pour les toutes premières fois qu'une attaque est observée.

3.3 SGNET : une plateforme de déploiement basée sur Scriptgen

Nous avons tiré parti de Scriptgen pour créer un réseau de pots de miel expérimental que nous avons appelé SGNET. SGNET suit les mêmes principes que Leurré.com mais fournit des données très significativement plus riches que celles offertes par ce premier système.

SGNET et l'étape d'injection de code SGNET est un environnement qui est prévu pour pouvoir passer à l'échelle et qui offre pratiquement le même niveau d'information qu'un pot de miel haute interaction pour une classe spécifique d'attaques, à savoir les attaques d'injection de code visant les serveurs et réalisées à l'aide de programmes déterministes. Nous sommes bien conscients que ceci ne couvre pas le spectre complet des attaques existantes. Cependant, encore aujourd'hui, cette classe d'attaques reste responsable de la création des grands botnets [32] et un des modes de propagation privilégiés d'un grand nombre de malware.

L'objectif final d'une attaque d'injection de code consiste à faire exécuter du code étranger à une machine victime en utilisant un service réseau vulnérable. Crandall et al. ont introduit dans [10] le modèle dit *epsilon-gamma-pi* qui décrit ce processus d'attaque selon trois phases distinctes :

Exploit (ϵ). Un ensemble de bytes envoyés sur le réseau correspondant à des données utilisées par l'application visée pour des décisions de contrôle de flot dans son arbre de décision. Cela correspond à un ensemble de requêtes que fait l'attaquant au service afin de l'amener dans un état où il peut mener l'attaque en tant que telle.

Données de contrôle erronées (γ). Un ensemble de bytes envoyés sur le réseau correspondant à des données utilisées par l'application visée pour contrôler son exécution et qui ont avoir pour effet de prendre le contrôle de son arbre d'exécution normal et rediriger l'application vers une zone mémoire différente de celle où se trouve son code légitime.

Payload (π). Un ensemble de bytes que l'attaquant veut faire exécuter par l'application visée en lieu et place de son code légitime et, ce, grâce aux phases préalables ϵ et γ.

Le payload à exécuter et qui fait partie de cette conversation réseau malveillante avec le service vulnérable est communément appelé le shellcode. Il est, la plupart du temps, limité à quelques centaines de bytes, voire moins. C'est pourquoi ses fonctionnalités sont très limitées et se restreignent la plupart du temps à récupérer, par upload ou download, un code binaire plus imposant : le malware lui même. Nous étendons donc le modèle original epsilon-gamma-pi en lui adjoignant une quatrième phase correspondant au malware obtenu après l'exécution du shellcode. Nous appelons cette nouvelle phase Mu (μ).

Une attaque peut être caractérisée par un quadruple $(\epsilon, \gamma, \pi, \mu)$. Les premiers vers étaient caractérisés par une relation simple entre ces phases : un ver était identifiable par un quadruple unique et la détection d'une des phases permettait de déduire les autres. Cette situation a changé. La disponibilité d'outils tels que Metasploit [37,33] permet à des utilisateurs non expérimentés de générer simplement leurs propres

variations de shellcode ou de malware tout en réutilisant le code nécessaire pour réaliser les autres phases. Il en résulte une explosion des combinaisons des phases observées. A un μ donné peut correspondre un grand nombre de tuples (ϵ, γ, π) de même qu'à un ϵ peut correspondre un grand nombre de variations de tuples (γ, π, μ).

Scriptgen, avec sa machine à états finis, nous offre un moyen efficace de reconnaître les différents phases ϵ. SGNET a été conçu de telle sorte qu'il puisse également nous donner des informations sur les autres phases et, ce, grâce à l'intégration d'autres outils. En particulier, nous tirons parti du mécanisme de détection de prise de contrôle d'un processus, offert par Argos [28] pour identifier les attaques d'injection de code qui réussissent, identifier les données de contrôle erronées γ et retrouver la position du shellcode π dans le flux de bytes envoyés sur le réseau. Par ailleurs, nous utilisons la capacité qu'a Nepenthes [2] d'émuler des shellcodes inconnus pour savoir comment aller récupérer le malware μ.

Le déroulement d'une attaque au sein de l'environnement SGNET se fait en plusieurs phases qui correspondent aux étapes décrites précédemment. SGNET distribue le traitement de ces phases à trois entités fonctionnelles distinctes : *la sonde*, *l'oracle*, *le gestionnaire de shellcodes*.

La sonde SGNET est l'interface entre SGNET et l'Internet. Comme pour le projet Leurré.com, nous voulons, dans le cadre de SGNET, observer un nombre limité d'adresses IPs dans une grande diversité d'endroits afin de pouvoir caractériser le plus fidèlement possible l'hétérogénéité des attaques observables [12,9]. C'est pourquoi les sondes SGNET sont des entités logicielles très légères, déployables à bas coût par tout partenaire désireux de rejoindre le projet. Chaque sonde observe les activités liées à un nombre très restreint d'adresses IPs (3). Grâce à Scriptgen, les sondes sont capables de gérer de façon autonome la première phase, ϵ, des attaques qui ont déjà été observées par le passé et dont le traitement a été automatisé grâce à la machine à états finis dans la sonde elle même.

SGNET dispose également d'un *oracle* qui lui sert à obtenir des traces d'interaction entre les attaquants et de vrais systèmes afin de pouvoir raffiner son niveau de connaissance. C'est particulièrement utile lorsque de nouvelles attaques apparaissent. Cet *oracle* utilise une véritable machine dans laquelle tourne une machine virtuelle dont le fonctionnement est contrôlé en permanence en utilisant une technique dite de *memory tainting*. Ceci est réalisé grâce à *Argos*, présenté par Portokalidis et al. dans [28]. Argos marque toutes les cases mémoires dont le contenu est dérivé du contenu de paquets reçus du réseau. Dès le moment où Argos détecte que le système d'exploitation veut exécuter du code présent dans une case mémoire marquée, il interrompt l'exécution. Argos a été intégré au système SGNET qui peut, à tout

instant, lancer une machine virtuelle Argos configurée de façon similaire à l'hôte attaqué qui en fait la demande (profil de machine, adresse IP, services, DNS, etc.)

L'*oracle* fournit des informations sur la présence de code injecté (γ) et nous permet de retrouver la position, dans le flux réseau, du premier octet exécuté par l'hôte attaqué, ce que nous notons comme étant l'octet B_i du payload π. Nous avons développé une heuristique pour identifié le payload π dans le flux réseau en partant d' *indices* que nous donne l'oracle [17]. Ceci nous permet d'inclure dans le processus d'apprentissage de Scriptgen de l'information complémentaire, à savoir une marque identifiant un état terminal de la machine à états finis stipulant qu'il correspond à une étape réussie d'injection de code et indiquant où, dans le paquet reçu, se trouve le code injecté.

L'étape finale de la gestion des attaques d'injection de code est prise en charge par le gestionnaire de shellcode de SGNET. Chaque payload π identifié lors de l'interaction avec SGNET est soumis au shellcode handler. Il a été implémenté en réutilisant certaines briques fonctionnelles de Nepenthes [2]. Plus précisément, nous utilisons la partie de Nepenthes qui analyse le shellcode pour *comprendre* ce qu'il est censé faire, en fonction de quoi une libraire adéquate est utilisée pour télécharger le malware lui même.

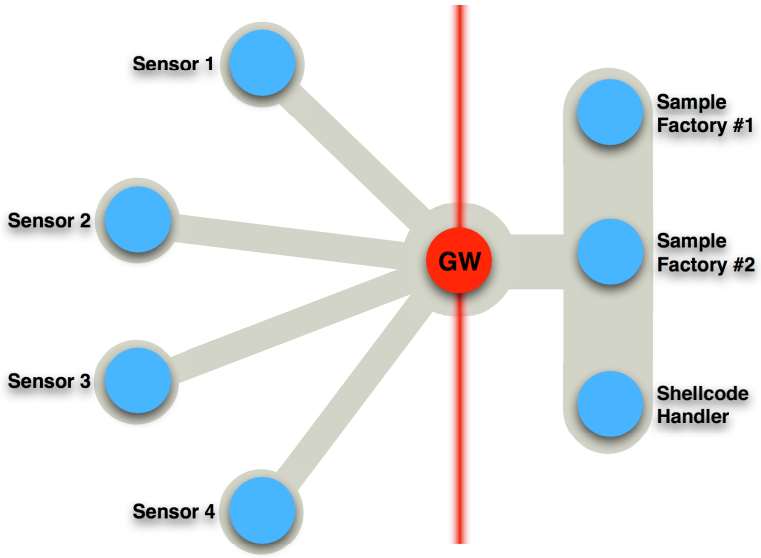


Fig. 6. Architecture SGNET

Architecture SGNET L'architecture générale SGNET est représentée dans la Figure 6. Toutes les entitées SGNET communiquent entre elles grâce à un protocole nommé Peiros [18] à la structure inspirée du protocole HTTP. Peiros permet les échanges sous forme de requêtes de services permettant, par exemple, à une sonde de demander l'instanciation d'un oracle au vol (noté *sample factory* sur la Figure). Les sondes, distribuées dans l'espace d'adressage IP et hébergées par différents partenaires du projeto, sont connectées à une entité centrale nommée la passerelle SGNET qui agit comme un proxy au niveau applicatif. La passerelle reçoit les requêtes de service des sondes et les transmet aux entités disponibles, réalisant de la sorte un équilibrage de charge sur les oracles et gestionnaires de shellcode. L'architecture offre une séparation propre entre les sondes, qui sont des démons simples fonctionnant sur des machines bon marché, et les entités internes plus complexes et gourmandes en ressources.

Le mécanisme d'apprentissage de Scriptgen a besoin d'une grande variabilité dans les traces qui lui sont offertes pour pouvoir apprendre *correctement* l'interaction à émuler. L'architecture représentée sur la Figure 6 montre comment la position centrale de la passerelle nous permet de maximiser nos chances d'observer une grande diversité de traces. En effet, toutes les conversations entre les sondes et les oracles passent par la passerelle qui peut les observer. La passerelle est donc l'endroit idéal pour mettre en œuvre le processus d'apprentissage grâce à son point de vue unique. A chaque fois que la passerelle produit un nouveau chemin dans la machine à états finis, elle va mettre à jour toutes les sondes grâce aux liens permanents qu'elle maintient avec elles.

Un aspect important de l'apprentissage dans Scriptgen est la relation stricte entre la capacité d'apprentissage et la configuration des oracles. Si un service ne fonctionne pas sur un oracle, nous ne pourrons évidemment pas apprendre comment il est attaqué. Il est donc important de choisir avec soin la façon dont les oracles sont configurés.

De ce qui précède, il ressort clairement que le déploiement de SGNET, s'il en partage les grands principes, est différent de son prédécesseur, le projet Leurré.com. En effet, SGNET est une architecture bien plus complexe qui nous permet d'augmenter de façon significative le niveau d'interaction des pots de miel sans pour autant nécessiter de machines coûteuses pour les sondes. Cependant, grâce au mécanisme d'apprentissage incrémental, les sondes sont capables de traiter de plus en plus d'attaques, ne nécessitant le recours à la passerelle et à l'oracle que lors de l'apparition de nouvelles attaques.

4 Analyse d'Événements d'Attaque

4.1 Identification des Événements

Evénement d'Attaque : Définition Un *évènement d'attaque* est défini comme étant un ensemble de séries temporelles de Clusters qui ont une même « forme particulière » durant un intervalle de temps bien défini. Cet intervalle de temps possède souvent une durée de quelques jours, mais il peut parfois ne durer qu'un seul jour voire quelques heures pour des activités très ponctuelles.

L'existence même de ces évènements d'attaque démontrent déjà clairement une coordination des activités de différentes machines attaquantes. Il est à noter qu'un tel ensemble peut dans certains cas être un singleton, ce qui est le cas typiquement pour des activités ne durant qu'un seul jour (ou moins d'un jour). Pour illustrer ce concept d'évènement d'attaque, le graphique supérieur de la Figure 7 représente l'évènement 225 qui est constitué du Cluster 60332 (visant la porte 5900 TCP) et qui attaque sept plateformes différentes (5, 8, 11, ..., 31) à partir du jour 393 jusque 400. Le graphe inférieur de la même Figure 7 représente par contre l'évènement d'attaque 14 qui est constitué d'activités liées au Cluster 0, ayant lieu uniquement au jour 307 et provenant presque exclusivement d'Espagne.

Description de l'ensemble de données Afin de conserver un ensemble de données assez « propres » pour les expérimentations, nous avons sélectionné les traces réseaux observées sur 40 plateformes (sur 50) à notre disposition. Ces 40 plateformes ont toutes fonctionné durant plus de 800 jours pendant lesquels aucune plateforme n'a été hors-service plus de 10 fois. De plus, chaque plateforme a fonctionné de manière continue pendant au moins 100 jours. Toutes les plateformes ont été en fonctionnement pendant un minimum de 400 jours sur toute la période considérée.

Le nombre total de sources observées sur toute la période, groupées par jour et sur l'ensemble de ces 40 plateformes, sera noté par la série temporelle initiale TS.

Nous pouvons à présent éclater cette série temporelle par pays d'origine de ces sources[3]. Ceci nous donne 231 séries temporelles TS_X où le i^{eme} point de ces séries indique le nombre de sources, observées sur toutes les plateformes, qui proviennent donc du pays X. Nous représentons par TS_L1 l'ensemble de ces séries temporelles de niveau 1. Pour réduire le coût en calculs, nous ne garderons que les pays pour lesquels nous avons pu observer au moins 10 sources pour un jour donné, ce qui nous permet de nous concentrer sur 85 séries temporelles au lieu des 231 initiales.

[3] L'origine géographique, basée sur chaque adresse IP, est obtenue grâce au produit Maxmind. Cependant, certaines IPs ne peuvent être mises en correspondance avec un pays spécifique et ont donc une étiquette particulière, par exemple EU, A1, ...

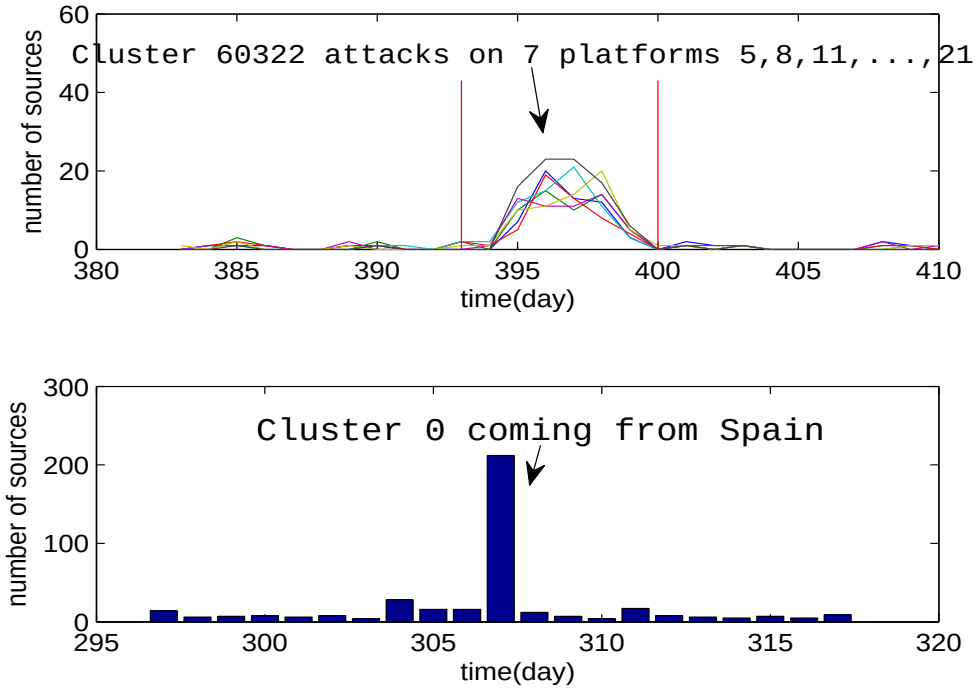


Fig. 7. Haut : le cluster 60232 attaque sept plateformes différentes entre le jour 393 et 400. En-dessous : un pic d'activité du cluster 0 venant principalement d'Espagne, au jour 307.

Tab. 4. Description des données : TS : *l'ensemble de toutes les sources observées durant la période considérée*, OVP : *point de vue pris pour l'observation*, TS_L1 : *ensemble de séries temporelles au niveau pays d'origine/plateforme visée*, TS_L1' : *ensemble de séries temporelles significatives dans* TS_L1, TS_L2 : *ensemble des séries temporelles par cluster*, TS_L2' *ensemble de séries temporelles par cluster qui présentent une variation suffisante*

TS contient 3,477,976 Sources		
OVP	pays	plateforme
$\|TS\_L1\|$	231	40
$\|TS\_L1'\|$	85 (94,4% de TS)	40 (100% de TS)
$\|TS\_L2\|$	436,756	395,712
$\|TS\_L2'\|$	2,420	2,127
sources	2,330,244 (67% de TS)	2,538,922 (73% de TS)

L'ensemble des pays correspondants sera dénoté par $big_{countries}$, et l'ensemble de séries temporelles de niveau 1 (à présent réduit) par TS_L1'. Ensuite, nous séparons chacune de ces séries temporelles par Cluster afin de produire l'ensemble final de séries temporelles $\Phi_{[0-800],c_i,country_j} \forall c_i$ et $\forall country_j \in big_{countries}$. Le i^{eme} point de la série temporelle $\Phi_{[0-800],X,Y}$ indique le nombre de sources provenant du pays Y qui ont été observées le jour i attaquant l'une de nos plateformes à l'aide d'une activité clairement définie par le cluster X. Nous représentons par TS_L2 l'ensemble de toutes ces séries temporelles de niveau 2. Dans ce cas, $|TS\_L2|$ est égal à 436.756, ce qui correspond à 3.284.551 sources. Tel qu'il est expliqué dans [27], les séries temporelles qui ne varient presque pas en amplitude durant les 800 jours sont quasiment inutiles pour identifier des évènements d'attaque intéressants, et donc nous pouvons également les éliminer. C'est pourquoi nous ne gardons que les séries temporelles qui présentent des variations suffisantes sur les 800 jours. Nous représentons cet ensemble raffiné de séries temporelles par TS_L2'. $|TS\_L2'|$ est finalement égal à 2.420, ce qui correspond à 2.330.244 sources.

De manière tout à fait similaire, nous avons appliqué la même procédure de nettoyage et de filtrage des traces en nous basant sur les plateformes visées, au lieu des pays d'origine. Les résultats correspondants sont donnés dans la Table 4.

Détection d'Évènements d'Attaque : Résultats Nous avons appliqué les techniques présentées dans [26] afin d'identifier les évènements d'attaque présents dans nos deux ensembles de données distincts, à savoir $TS_{country}$ et $TS_{platform}$. Pour les séries temporelles de $TS_{country}$ (resp. $TS_{platform}$), nous avons trouvé 592 (resp. 690) évènements d'attaque qui correspondent à 574.125 (resp. 578.372) sources. Les résultats sont donnés dans la Table 5.

Tab. 5. Détection d'évènements d'attaque : résultats

AE-set-I($TS_{country}$)		AE-set-II($TS_{platform}$)	
No.AEs	No.sources	No.AEs	No.sources
592	574,125	690	578,372

No.AEs : amount of attack events

4.2 Armées de Zombies

Jusqu'ici, nous avons identifié ce que nous avons appelé des évènements d'attaque, qui mettent en évidence l'existence d'attaques coordonnées lancées par un groupe de machines compromises, i.e. une armée de zombies. Il serait intéressant de vérifier si la même armée se manifeste également au travers de plusieurs évènements. Pour faire cela, nous proposons de calculer ce que nous appellerons les « ensembles d'actions ». Un *ensemble d'actions* est un groupe d'évènements d'attaque qui sont fort probablement générés par la même armée de zombie. Dans cette section, nous montrons comment reconstruire ces ensembles d'actions et quelle information nous pouvons dériver de ceux-ci concernant la taille et la durée de vie de ces armées de zombies.

Identification des armées Mesures de Similarité : Dans sa forme la plus simple, une armée de zombies est un botnet classique. Elle peut aussi être constituée de plusieurs botnets, c'est à dire plusieurs groupes de machines écoutant sur des canaux *C&C* distincts. Ceci est invisible pour nous, et finalement peu important. Ce qui importe, c'est que toutes ces machines agissent de manière coordonnée. Au fil du temps, il est raisonnable de s'attendre à ce que certains membres d'une armée soient « nettoyés » tandis que de nouveaux membres apparaissent. De ce fait, si la même armée attaque nos pots de miel deux fois mais à des périodes distinctes, une manière assez simple de relier ces deux évènements d'attaque consiste à observer qu'ils partagent un nombre d'adresses IP en commun qui est pas négligeable. De manière plus formelle, on mesure la probabilité que deux évènements e_1 et e_2 soient liés à la même armée de zombies au moyen de la mesure de similarité suivante :

$$sim(e_1, e_2) = \begin{cases} max(\frac{|e_1 \cap e_2|}{|e_1|}, \frac{|e_1 \cap e_2|}{|e_2|}) & \text{si } |e_1 \cap e_2| < 200 \\ 1 & \text{autrement} \end{cases}$$

où $|e_1|$ (resp. $|e_2|$) représente le nombre d'adresses IP distinctes de l'évènement d'attaque e_1 (resp. e_2) et $|e_1 \cap e_2|$ représente le nombre d'adresses communes aux deux évènements d'attaque e_1 et e_2. On conclut que e_1 et e_2 ont été causés par la même armée de zombies si et seulement si $sim(e_1, e_2) > 10\%$.

Ensemble d'Actions : Nous pouvons à présent utiliser la fonction $sim()$ pour regrouper les évènements d'attaque en « ensembles d'actions ». Pour cela, nous construisons un simple graphe dans lequel les noeuds sont les évènements d'attaque. Il existe un arc entre deux noeuds e_1 et e_2 si et seulement si $sim(e_1, e_2) > \delta$. Tous les noeuds qui sont connectés par au moins un chemin se retrouve alors dans le même ensemble d'actions. En d'autres termes, nous avons autant d'ensembles d'actions que nous avons de graphes déconnectés qui comportent au minimum deux noeuds (les singletons ne sont pas considérés comme ensembles d'actions).

Notons que notre approche est telle que nous pouvons identifier un ensemble d'actions composé de trois évènements d'attaque e_1, e_2 et e_3, où $sim(e_1, e_2) > \delta$ et $sim(e_2, e_3) > \delta$ mais par contre $sim(e_1, e_3) < \delta$. Ceci est tout à fait cohérent avec notre intuition que les armées puissent évoluer dans le temps, de telle manière que les machines d'une certaine armée peuvent être assez différentes de celles identifiées lors de l'observation précédente.

Résultats : Nous avons pu identifier 40 (resp. 33) armées de zombies à partir du AE-set-I (resp. AE-set-II), qui ont réalisé un total de 193 (resp. 247) évènements d'attaque. La Figure 8 représente la distribution des évènements d'attaque par armée de zombies. Le graphe du haut (resp. du bas) représente la distribution obtenue à partir du AE-set-I (resp. AE-set-II). Nous pouvons voir que le nombre maximum d'évènements pour une armée est de 53 (resp. 47) tandis que 28 (resp. 20) armées n'ont pu être observées que deux fois.

Caractéristiques Principales des Armées de Zombies

Durée de vie La Figure 9 représente la distribution cumulative de la durée de vie minimum des armées de zombies obtenues à partir de $TS_{platform}$ et $TS_{country}$ (voir Section 4.1). Selon ce graphique, environ 20% des armées ont existé pendant plus de 200 jours. Dans les cas extrêmes, deux armées semblent avoir survécu pendant 700 jours ! Un tel résultat semble indiquer que soit i) cela prend un temps considérable pour nettoyer toutes ces machines compromises, ou que soit ii) certaines armées sont capables de rester actives pendant de longues périodes de temps, malgré le fait que certains de leurs membres disparaissent, et sans doute grâce à l'infection de nouvelles machines pour les remplacer.

Durée de Vie des Hôtes Infectés Nous pouvons classer les armées en deux classes, tel que mentionné dans la section précédente. Par exemple, la Figure 4.2a représente la matrice de similarité de l'armée 33 (ou ZA33). Pour construire cette matrice, nous avons d'abord ordonné les 42 évènements d'attaque qui la composent selon la date à laquelle ils se sont produits. Ensuite, nous représentons toutes les relations de

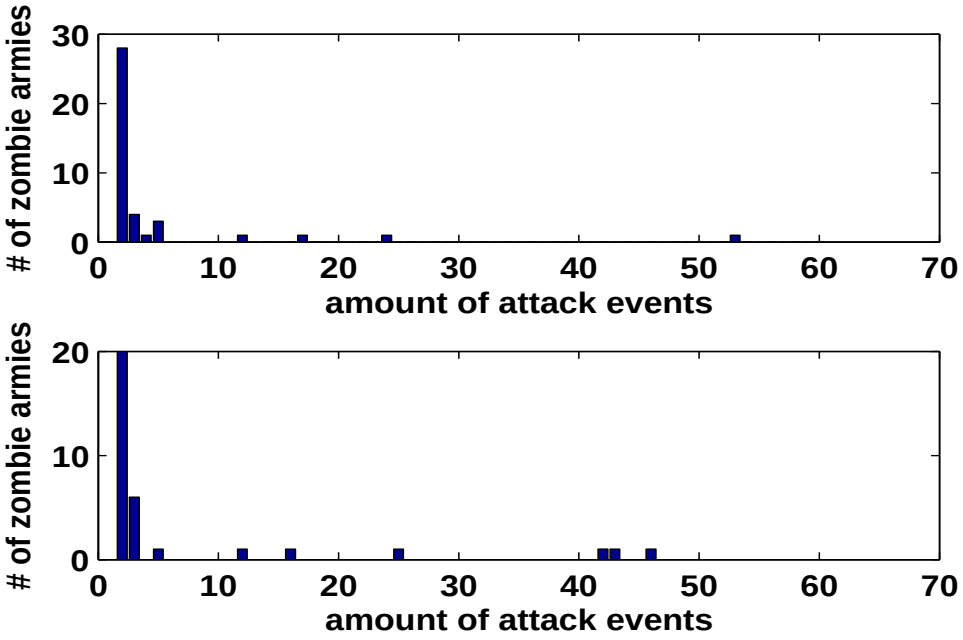


Fig. 8. Taille des armées de zombies

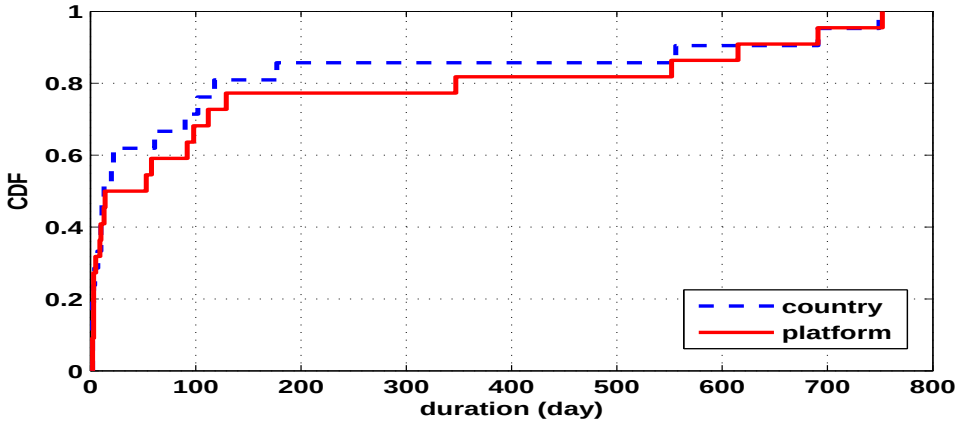
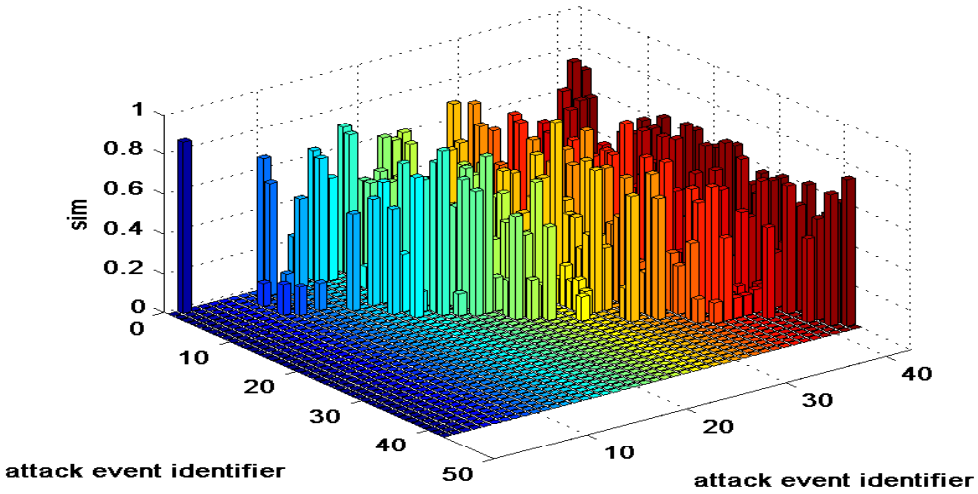
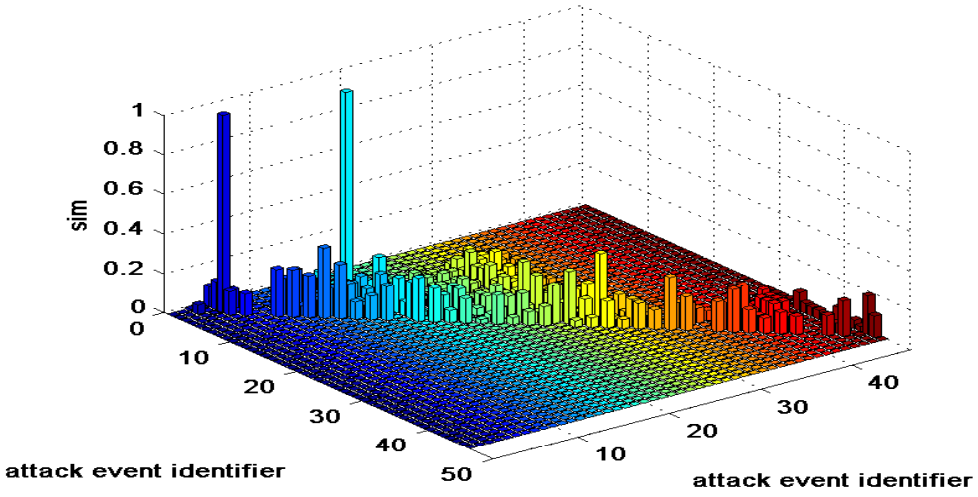


Fig. 9. CDF de la durée de vie



(a)



(b)

Fig. 10. Taux de renouvellement des armées de zombies

similarité sous la forme de cette matrice de similarité $\mathscr{M}$ aux dimensions 42×42. La cellule (i,j) représente donc la valeur de $sim()$ entre les évènements d'attaque ordonnés i et j. Puisqu'il s'agit d'une matrice symétrique, nous ne représentons que la moitié de celle-ci pour des raisons de lisibilité.

Tel qu'on peut le voir, nous obtenons des mesures de similarité assez élevées entre quasiment tous les évènements d'attaque (i.e., environ 60 %). Ceci est également vérifié entre le tout premier et le dernier évènement. Il est important de noter que l'intervalle de temps entre le début et la fin des activités de cette armé est de 753 jours !

La Figure 4.2b représente le cas opposé avec l'armée de zombie 31 (ou ZA31), qui est constituée de 46 évènements d'attaque. Nous procédons de la même manière que ci-dessus pour construire la matrice de similarité. On peut y voir que les valeurs élevées sont toutes situées autour de la diagonale de $\mathscr{M}$. Ceci signifie que l'évènement d'attaque i possède le même sous-ensemble de machines infectées que les quelques autres évènements se produisant dans un intervalle de temps assez proche. Il est aussi important de noter que cette armée change ses vecteurs d'attaque au fil du temps. En fait, cette armée modifie constamment le type d'attaque en passant de 4662 TCP, à 1025 TCP, puis 5900 TCP, 1443 TCP, 2967 TCP, 445 TCP,... La durée de vie de cette armée est de 563 jours. En se basant sur ces deux exemples, il apparaît donc que la composition des armées évolue dans le temps de différentes manières. Des recherches supplémentaires doivent encore être menées afin de mieux comprendre les raisons stratégiques sous-jacentes.

4.3 Impact du Point d'Observation

Analyse La Table 5 met en évidence le fait que la manière de décomposer l'ensemble initial de traces d'attaque (i.e. les séries temporelles initiales TS) influence la composition finale des évènements d'attaque considérés. Le point d'observation, à savoir la décomposition soit par pays d'origine, soit par plateformes visées, a donc un impact sur l'identification des évènements. Afin d'évaluer le recouvrement entre ces différents ensembles d'évènements, nous utilisons la rapport de sources communes, ou *common source ratio, ou csr*, défini comme étant :

$$csr(e, AE_{op'}) = \frac{\sum_{\forall e' \in AE_{op'}} |e \cap e'|}{|e|}$$

où $e \in AE_{op}$ et $|e|$ est le nombre de sources présentes dans l'évènement e, AE_{op} est *AE-set-I* et $AE_{op'}$ est *AE-set-II* (ou vice versa).

La Figure 11 représente les deux fonctions de distribution cumulatives correspondant à cette mesure. Le point (x, y) sur la courbe signifie qu'il y a $y * 100\%$

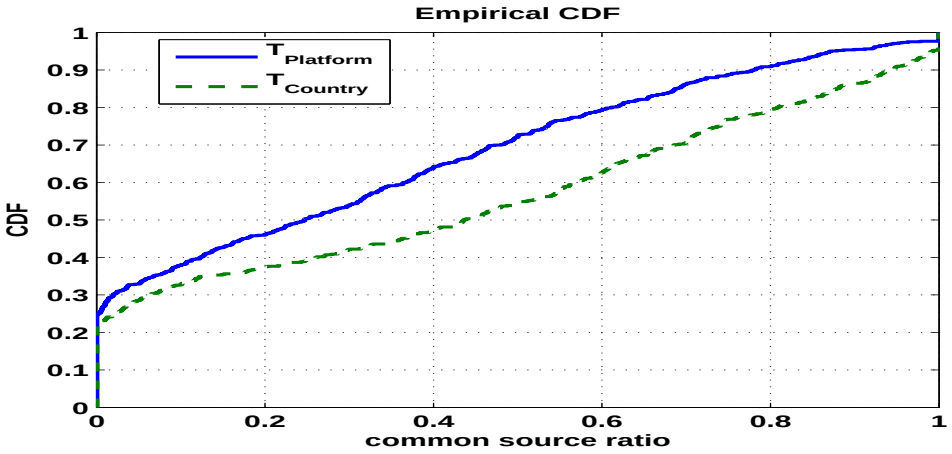


Fig. 11. CDF du ratio de sources communes

d'évènements d'attaque obtenus à partir de $T_{country}$ (resp $T_{platforms}$) qui ont moins de $x * 100\%$ de sources en commun avec tous les évènements d'attaque obtenus à l'aide de $T_{platforms}$ (resp $T_{country}$).

La courbe $T_{country}$ représente donc la fonction de distribution cumulative (CDF) obtenue dans ce premier cas et la courbe $T_{platforms}$ représente la CDF obtenue à partir des évènements d'attaque qui composent l'ensemble de séries temporelles $T_{platforms}$. Tel qu'on peut le remarquer, environ 23% (resp. 25%) des évènements d'attaque obtenus à partir de $T_{country}$ (resp. $T_{platform}$) ne partagent aucune source en commun avec les évènements identifiés en partant des séries temporelles $T_{platform}$ (resp. $T_{country}$). Ceci correspond à 136 évènements (16.919 sources) et 171 évènements (75.920 sources) respectivement. Au total, il y a 288.825 (resp. 293.132) sources présentes dans AE-Set-I (resp. AE-Set-II), mais pas dans AE-Set-II (resp. AE-Set-I).

Discussion Il y a de bonnes raisons pour lesquelles nous ne pouvons pas considérer uniquement un seul point de vue pour détecter tous les évènements d'attaques. Nous développons quelques unes de ces raisons ci-dessous.

Séparation par pays : Supposons que nous ayons un botnet B composé de machines situées dans les pays $\{X, Y, Z\}$. Supposons aussi que, de temps à autre, ces machines attaquent nos plateformes et laissent des traces qui sont attribuées au cluster C. Supposons ensuite que ce cluster C soit fort « populaire », c'est à dire qu'un bon nombre d'autres machines dans le monde laissent des traces similaires de manière continue sur nos plateformes. Ainsi, les activités liées au botnet B sont noyées

dans le bruit de fond provenant de toutes les autres machines assignées également au cluster C. C'est particulièrement vrai pour la série temporelle du cluster C (tel que défini précédemment), et ça peut aussi être le cas pour les séries temporelles obtenues en séparant par plateforme, $\Phi_{[0-800],C,platform_i} \forall platform_i \in 1..40$. Néanmoins, en séparant les séries temporelles correspondant au cluster C par pays d'origine des sources, il est fort probable que les séries temporelles $\Phi_{[0-800],C,country_i} \forall country_i \in \{X, Y, Z\}$ seront fortement corrélées durant les périodes d'activité de ce botnet situé dans les pays en question. Ceci permettra donc d'identifier un ou plusieurs évènements d'attaque lié à ce botnet.

Séparation par plateforme : De façon duale, supposons que nous ayons un botnet B' composé de machines situées partout dans le monde. Supposons aussi que, de temps à autre, ces machines attaquent un sous-ensemble spécifique de plateformes $\{X, Y, Z\}$ et laissent des traces assignées au cluster C. Supposons ensuite que ce cluster C soit assez populaire, et que donc un bon nombre d'autres machines dans le monde laissent des traces similaires de façon continue sur toutes nos plateformes. Il en résulte alors que les activités spécifiquement liées au botnet B' seront perdues dans le bruit généré par toutes les autres machines liées au même cluster C. Ceci est certainement vrai pour la série temporelle du cluster C (tel que défini précédemment), mais aussi pour les séries temporelles obtenues en séparant par pays d'origine, $\Phi_{[0-800],C,country_i} \forall country_i \in big_{countries}$. Néanmoins, en séparant maintenant les séries temporelles du cluster C par plateforme attaquée, il est fort probable que les séries temporelles $\Phi_{[0-800],C,platform_i} \forall platform_i \in \{X, Y, Z\}$ seront fortement corrélées durant les périodes d'activité de ce botnet où seules certaines plateformes sont particulièrement visées. Ceci permettra donc d'identifier un ou plusieurs évènements d'attaque lié à ce botnet.

Le graphique du haut de la Figure 12 représente l'évènement d'attaque 79. Dans ce cas précis, nous voyons que les traces liées au cluster 175309 sont fortement corrélées quand les traces sont groupées par plateforme visée. Il y a en fait 9 plateformes visées dans ce cas-ci, impliquant 870 sources. Si nous groupons le même ensemble de traces par pays d'origine, nous obtenons les courbes du graphique inférieur de la Figure 12 où l'évènement d'attaque identifié précédemment est à peine visible. Ceci met en évidence l'existence d'un botnet composé de machines situées partout dans le monde, et qui vise un sous-ensemble de réseaux spécifiques sur Internet.

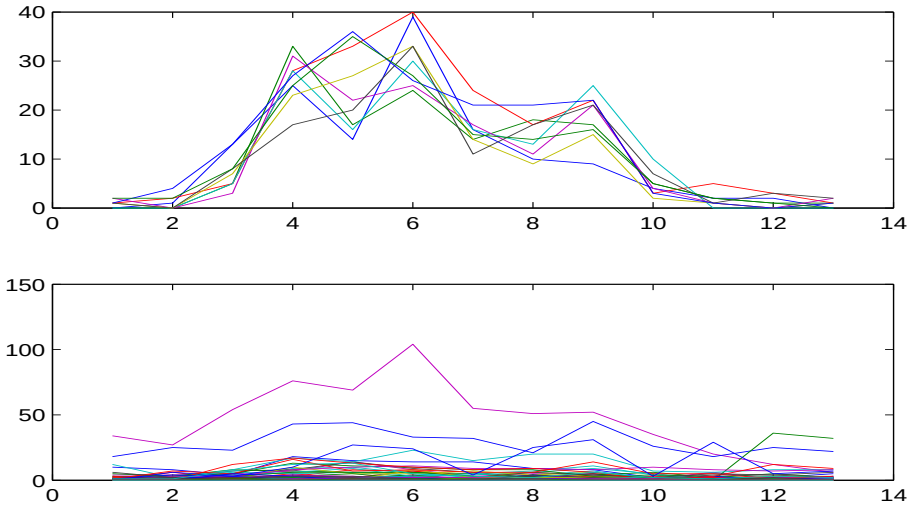


Fig. 12. En-haut : l'évènement d'attaque 79 lié au cluster 175309 sur 9 plateformes visées. En-dessous : évolution du même cluster mais par pays d'origine. Le bruit des attaques visant les autres plateformes diminue fortement le niveau de corrélation des séries temporelles du cluster séparé par pays d'origine.

5 Analyse Multi-Dimensionnelle des Événements d'Attaque

5.1 Méthodologie

Comme dans les enquêtes criminelles, un analyste en sécurité a besoin de synthétiser différents indices afin d'identifier les causes fondamentales probables qui sont à l'orgine des phénomènes globalement distribués sur Internet. Cette tâche est loin d'être triviale et peut se révéler être longue et fastidieuse. Par ailleurs, ce processus d'analyse dépend fortement du niveau d'expertise de l'analyste et il peut impliquer un bon nombre de caractéristiques (ou dimensions) différentes liées aux attaques. Ceci nous a amené à développer une méthodologie multi-dimensionnelle de découverte de connaissances (« knowledge discovery ») et de fouille de données (« data mining ») qui puisse nous aider à améliorer, de façon plus systématique, la compréhension des phénomènes d'attaque et des nouvelles menaces sur Internet. L'objectif ultime consiste à obtenir un système de cyber-alerte (« cyber situational awareness ») efficace, nous permettant d'évaluer la situation actuelle du niveau des menaces présentes sur Internet.

L'idée générale consiste à *i)* extraire des éléments d'information en fouillant un ensemble d'évènements selon différentes dimensions liées aux caractéristiques pertinentes de ces attaques ; *ii)* combiner systématiquement ces éléments d'information

afin de créer des concepts de plus haut niveau qui puissent mieux mettre en valeur les causes fondamentales à l'origine du trafic malveillant. Nous développons chacune de ces étapes ci-dessous.

5.2 Regroupement par Cliques

Principes Le premier composant de notre méthodologie de découverte d'information implique tout d'abord un processus de regroupement (ou de classification) non-supervisé, s'appuyant sur une technique de représentation par graphes qui vise à regrouper des évènements similaires ou fortement liés de manière fiable et cohérente. Pour cela, nous utiliserons les vecteurs caractéristiques obtenus à partir de ces évènements d'attaque.

Typiquement, les tâches de classification impliquent les étapes classiques suivantes : [16] : *i)* sélection et extraction d'une caractéristique (« feature »), suivi de la représentation du motif extrait ; *ii)* définition d'une mesure de similarité entre les paires de motifs ; *iii)* regroupement à proprement dit des motifs similaires ; *iv)* représentation compacte et abstraite (si nécessaire) de chaque groupe obtenu (ou « cluster ») ; et *v)* évaluation de la qualité et de la cohérence de ces groupes (si nécessaire également).

Dans n'importe quel type de classification, nous devons d'abord sélectionner les caractéristiques pertinentes qui pourront nous fournir des *motifs* (ou « patterns ») représentatifs à partir des données brutes (dans notre cas, il s'agit des évènements d'attaque). Ces motifs caractéristiques sont alors représentés à l'aide de *vecteurs caractéristiques* (ou « feature vectors »), qui pourraient être par exemple dans notre cas les distributions géographiques des attaquants, les distributions de réseaux d'origine, les séries temporelles ou encore la répartition des plateformes visées.

Dans un second temps, nous devons définir une mesure de similarité appropriée entre deux motifs. Pour cela, différents types de distances et de mesures de similarité existent dans la littérature scientifique (par exemple corrélation au sens de Pearson, Minkowski, Jackknife, etc.), mais il est clair que le choix de cette distance est assez crucial et aura un impact important sur les propriétés finales des groupes obtenus après le clustering (taille des clusters, cohérence, etc). Afin de comparer de manière cohérente des distributions de probabilités empiriques (telles que celles qu'on peut extraire de nos évènements d'attaque), nous nous appuyons sur des distances statistiques assez fortes qui sont dérivées de tests statistiques non-paramétriques (puisqu'on ne connaît, a priori, pas le type de distribution), tels que Pearson's χ^2 et Kolmogorov-Smirnov. La valeur p obtenue par ces tests est alors validée à l'aide de la divergence de Kullback-Leibler, qui est une distance statistique assez populaire en théorie de l'information. Toutes ces méthodes permettent finalement de déterminer si deux distributions de

probabilités diffèrent fortement ou si on peut considérer qu'elles proviennent d'une même population (et donc qu'il n'y aucune différence significative entre les deux distributions).

Finalement, nous profitons de ces mesures de similarité pour regrouper ensemble tous les évènements d'attaque dont les motifs (c'est à dire les distributions) sont fort similaires (au sens statistique). Nous utilisons simplement une approche basée sur les graphes afin de formuler ce problème : les noeuds du graphe représentent en fait les motifs des évènements, et les arcs expriment les relations de similarité entre chaque paire de motifs. Ensuite, le regroupement est effectué en identifiant dans le graphe des *cliques de poids maximum* (ou « maximal weighted cliques », MWC). Une clique maximale est définie comme étant un sous-graphe induit dans lequel les noeuds sont tous interconnectés, et la clique maximale ne peut pas être incluse dans une autre clique. Une clique de poids maximum ajoute encore une contrainte sur la somme de tous les poids liés aux arcs de la clique (qui doit donc aussi être maximalisée). Etant donné qu'il s'agit d'un problème de type *NP-complet* [4], plusieurs algorithmes approximatifs ont été proposés pour résoudre ce problème efficacement. Le lecteur intéressé est invité à consulter [39,38] pour une description plus détaillée de cette technique de regroupement par cliques appliquée aux traces d'un honeynet.

Quelques Résultats Expérimentaux. Nous avons appliqué notre méthode de regroupement par cliques à un ensemble de données provenant d'un honeynet, et composé de 351 évènements d'attaque pour un total de 282.363 sources IP (collectés sur Internet dans une période allant de septembre 2006 à juin 2008). Ces évènements ont été observés sur 36 plateformes situées dans 20 pays différents, et appartenant à 18 réseaux de Classe A différents. En terme d'activités réseau, toutes les sources ont pu être classées dans seulement 136 clusters.

La Table 6 présente un aperçu des cliques obtenues pour chaque dimension séparément. Comme on peut le voir, un proportion importante de ces sources a pu être classée dans des cliques. Le nombre élevé de corrélations par rapport aux séries temporelles des évènements d'attaques suggère qu'une majorité de ces évènements ont été en fait coordonnés, ou en tous cas synchronisés sur plusieurs pots de miel. Parmi les séquences de porte visés, on observe principalement des ports assez couramment utilisés et attaqués (par exemple les ports Windows utilisés par SMB ou RPC, les ports SQL, VNC, ...), mais aussi un très grand nombre de ports TCP élevés qui sont normalement fermés ou non utilisés sur des machines « standards » (ou disons, non-infectées). Un volume considérable de sources est également dû à des spammers UDP visant le service pop-up lié à Windows Messenger (ports 1026-1028 UDP).

Tab. 6. Un aperçu des résultats expérimentaux de cliques obtenus à partir d'un ensemble de données honeynet (+/- 1 an de trafic).

Dimension d'attaque	Nr de Cliques	Volume de sources (%)	Séquences de portes les plus visées
Géolocalisation	45	66.4	1027U, I, 1433T, 1026U, I445T, 5900T, 1028U, 9763T, I445T80T, 15264T, 29188T, 6134T, 6769T, 1755T, 64264T, 1028U1027U1026U, 32878T, 64783T, 4152T, 25083T, 9661T, 25618T, ...
Subnets IP (Classe A)	30	56.0	1027U, I, 1433T, 1026U, I445T, 5900T, 1028U, 9763T, 15264T, 29188T, 6134T, 6769T, 1755T, 50656T, 64264T, 1028U1027U1026U, 32878T, 64783T, 18462T, 4152T, 25083T, 9661T, 25618T, 7690T, ...
Plateformes visées	17	70.1	I, 1433T, I445T, 1025T, 5900T, 1026U, I445T139T445T139T445T, 4662T, 9763T, 1008T, 6211T, I445T80T, 15264T, 29188T, 12293T, 33018T, 6134T, 6769T, 1755T, 2968T, 26912T, 50656T, 64264T, 32878T, ...
Séries temporelles	82	92.2	135T, I, 1433T, I445T, 5900T, 1026U, I445T139T445T139T445T, I445T80T, 6769T, 1028U1027U1026U, 50286T, 2967T, ...

Visualisation des Cliques d'Attaquants. Afin d'évaluer à présent la qualité et la cohérence des cliques d'évènements d'attaque, il serait utile de pouvoir les représenter à l'aide d'un graphique ou d'une « carte » (par exemple à deux dimensions pour faciliter une évaluation visuelle). Le but de cette étape est de : *i)* vérifier les proximités entre les membres d'une même clique (cohérence intra-clique), et *ii)* relever d'éventuelles relations entre différentes cliques assez proches (c'est à dire la relations inter-clique). Il est à noter qu'une clique peut être considérée comme la définition la plus stricte d'un cluster. De plus, les mesures statistiques utilisées pour former ces cliques les rendent intrinsèquement fort cohérentes, ce qui signifie que certaines cliques, bien que différentes, peuvent être tout de même liées au même phénomène réel mais qui a sans doute évolué dans le temps entre les différentes observations (cf Section 4.2).

Etant donné que les vecteurs caractéristiques (par exemple les distributions statistiques) auxquels nous sommes confrontés ont un grand nombre de variables (typiquement, un vecteur géographique représente plus de 200 pays d'origine, chaque pays étant considéré comme une variable), la structure d'un tel ensemble de données n'est pas évidente à représenter en 2D, vu le nombre de dimensions. Les techniques appelées *Multidimensional scaling* (MDS) permettent de contourner ce problème en réduisant le nombre de dimensions à représenter, tout en conservant la même structure sous-jacente présente dans les données originales non-réduites. Par conséquent, ce genre de techniques MDS permette à l'analyste de visualiser les proximités entre les observations ou les objets analysés.

Les données collectées à partir de phénomènes réels sont intrinsèquement fortement non-linéaires, c'est pourquoi nous avons choisi une technique MDS développée récemment, appelée *t-SNE*, afin de visualiser les cliques dans chaque dimension d'attaque et évaluer leur pertinence. t-SNE [40] est une variante de*Stochastic Neighbour Embedding* (SNE) ; elle produit des visualisations 2D nettement meilleures que d'autres techniques MDS en réduisant la tendance à grouper fortement trop de points

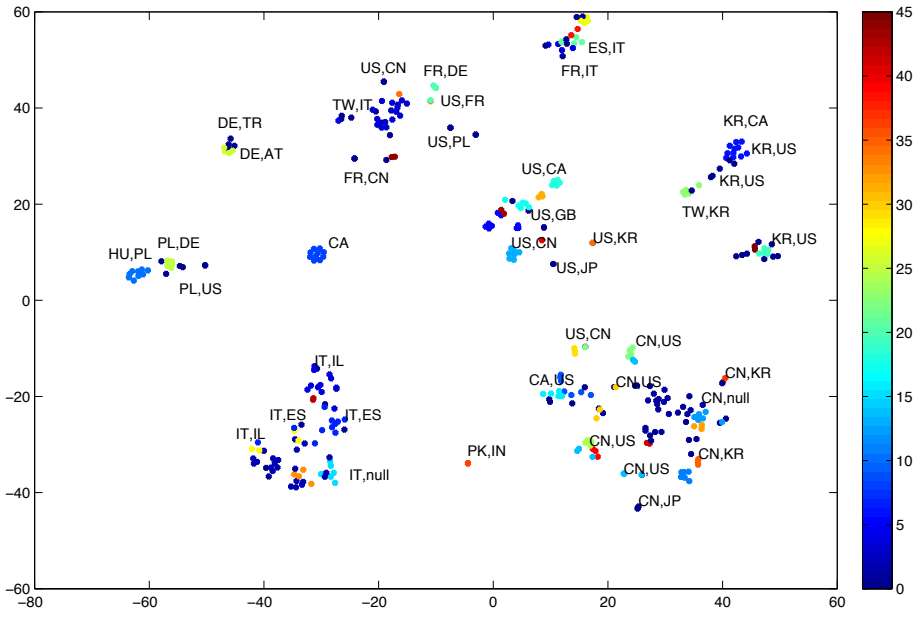


Fig. 13. Visualisation de cliques géographiques d'attaquants. L'échelle de couleur correspond aux identifiants des différentes cliques. Le texte superposé aux points indiquent les deux plus importants pays d'origine des attaquants.

au centre de la carte. De plus, cette technique a prouvé sa capacité et son efficacité à retenir à la fois la structure locale et globale des données réelles non-réduites, en comparaison avec d'autres techniques non-linéaires de réduction de dimensions telles que Sammon mapping, Isomaps ou Laplacian Eigenmaps.

La Figure 13 montre la carte 2D obtenue en réduisant les vecteurs géographiques à seulement deux dimensions en utilisant t-SNE. Chaque point sur ce graphique représente donc un vecteur de pays d'origine (c'est à dire une distribution) d'un évènement d'attaque particulier. L'échelle de couleur indique l'appartenance à une clique déterminée, ce qui est le résultat du regroupement par cliques effectué précédemment. On peut facilement vérifier que deux évènements adjacents sur ce graphique ont des distributions de pays fortement similaires (même d'un point de vue statistique), tandis que deux évènements éloignés n'ont absolument rien en commun au niveau de la caractéristique considérée (par exmple, le pays d'origine, etc). Il est assez surprenant d'ailleurs de constater que la mapping est loin d'être chaotique ; sa structure présente en effet pas mal de groupements de points assez clairs, ce qui traduit de fortes relations entre plusieurs groupes d'attaques en ce qui concerne leurs origines. Par l'utilisation même de distances statistiques très strictes, il est quasiment inconcevable que ce résultat puisse être dû uniquement au hasard.

De façon similaire, d'autres types de *mapping sémantique* peuvent être obtenus pour les autres dimensions d'attaque considérées (par exemple par subnets d'origine, plateformes visées, etc) afin d'aider à évaluer la qualité des regroupements par cliques des évènements d'attaque. Pour conclure cette illustration, la Figure 14 montre le même mapping géographique sur lequel on a superposé les séquences de ports pour différents points. Cette Figure donne une autre perspective tout à fait intéressante puisqu'elle permet de visualiser des relations à priori non-évidentes entre les différents types d'activités et leurs origines. Ceci nous suggère donc qu'il serait extrêmement intéressant de pouvoir *combiner différentes cliques* d'attaquants, afin d'en extraire des informations encore plus riches sur les phénomènes qui ont causé ce trafic malveillant. Cet aspect est introduit dans la section suivante.

5.3 Combinaison de Cliques d'Attaquants

Le deuxième composant de notre méthodologie d'analyse est conceptuellement similaire à un processus de fusion de données. A partir des différents ensembles de cliques obtenus, l'idée consiste à combiner k des N dimensions d'attaque, avec $k = 2, ..., N$, afin de découvrir des connaissances plus raffinées sur certains phénomènes et leurs causes fondamentales (par exemple, une série d'évènements d'attaque causés par le même botnet ou la même famille de vers, l'évolution d'un botnet par rapport aux subnets IP, etc). Chaque clique contient déjà en soi de l'information intéressante

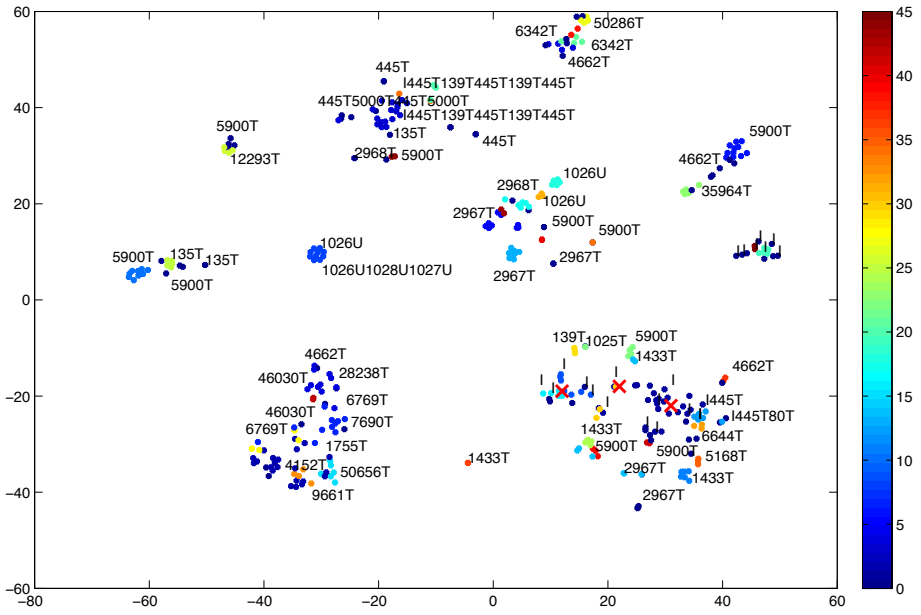


Fig. 14. Même visualisation des cliques géographiques qu'à la Fig 13, mais les étiquettes superposées indiquent à présent les séquences de ports réalisées par les attaquants.

à propos d'un certain aspect lié aux phénomènes d'attaque observés. Mais dans la plupart des cas, un analyste devra synthétiser différents indices afin de déterminer exactement les causes d'un phénomène et éventuellement comprendre avec plus de précision ce qui s'est passé. C'est pourquoi nous pouvons profiter de toutes les cliques obtenues pour chaque dimension pour construire des concepts de plus haut niveau en combinant tout simplement différentes cliques (par exemple par des intersections de cliques). En fonction du phénomène à étudier, l'analyste devra alors ajouter plus ou moins de caractéristiques, et donc ajouter de la sémantique en combinant autant de cliques qu'il est nécessaire pour mieux comprendre le phénomène. En pratique, on observe en fait que le nombre de concepts ou de combinaisons n'est pas excessif. En conclusion, alors que l'analyse manuelle de traces réseau « brutes » composées de millions de paquets serait forcément extrêmement compliquée, nous avons à présent un outil assez puissant qui permet d'extraire directement un nombre limité de concepts combinés qui donnent une bien meilleure perspective (plus concise et plus descriptive) sur les phénomènes réels qui ont causé ce trafic.

Illustration de l'analyse multi dimensionnelle Lorsque nous appliquons l'analyse multi dimensionnelle aux exemples donnés dans 2.4 nous découvrons que ces évènements d'attaque participent d'un phénomène de grande ampleur que nous supposons être lié à l'activité d'un botnet. Ce phénomène a été actif dans une période s'écoulant du 1er décembre 2006 jusqu'au 31 mars 2007, durant laquelle nous avons observé 67 évènements d'attaque qui peuvent être regroupés en quatre vagues distinctes (voir Fig 15) en fonction de leur dimension temporelle. Les corrélations par plateforme indiquent que tous ces évènements ont touché exactement le même ensemble de plateformes (principalement en Belgique et au Royaume Uni). En ce qui concerne l'origine des attaques (pays et sous réseaux d'appartenance des sources), notre méthode met en lumière deux communautés distinctes d'attaquants : un grand groupe de *découvreurs* (se bornant à faire des scans ICMP sur toutes les adresses des IPs des plateformes) et un groupe plus petit *d'exploitants* (réalisant une attaque réelle sur les ports 445T ou 139T suite à l'envoi d'un paquet ICMP). Il est intéressant de noter que ces derniers semblent *connaître* quels pots de miel émulent une machine Windows car ils attaquent presque exclusivement les adresses IPs correspondant à celles ci. Enfin, l'évolution de la population des botnets, en termes de blocs d'adresse IPs, entre chaque vague d'attaque, est visible à partir du mapping des différentes cliques des attaquants. Elle révèle un dernier élément intéressant. La communauté des *découvreurs* a été éclatée en plusieurs cliques mais qui toutes appartiennent à la même région géographique (comme le montrent les trois croix rouges dans le coin inférieur droit de la carte présentée dans la Figure 14)

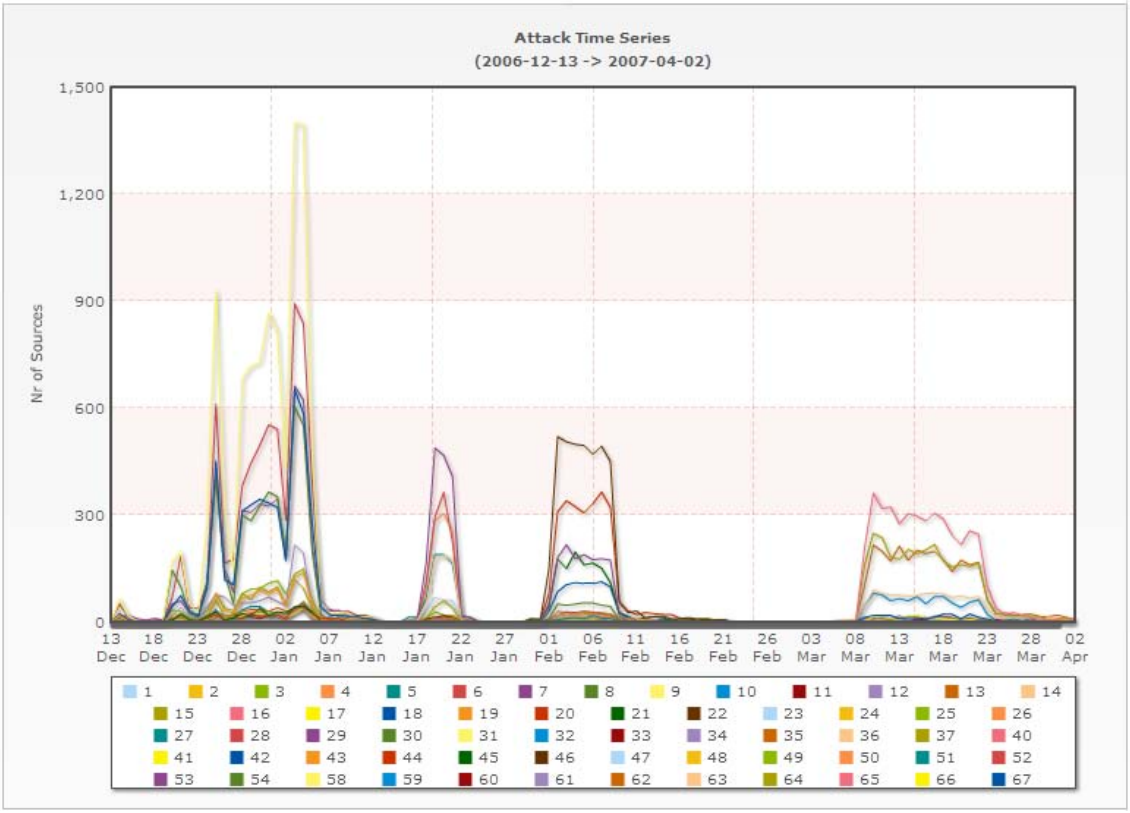


Fig. 15. Séries temporelles associées à un phénomène de grande ampleur lié à l'activité d'un botnet, composées de 67 évènements d'attaques observées entre le 6 Décembre et le 7 avril 2008.

6 Au delà de la corrélation d'évènements : analyse de l'espace $(\epsilon, \gamma, \pi, \mu)$

Dans les sections précédentes, nous avons tiré parti de techniques de corrélation pour analyser l'information à notre disposition et en inférer des faits significatifs sur l'identité et le comportement des clients responsables d'évènements observés. Nous avons retardé jusqu'ici l'analyse des *effets* de ces activités sur les victimes. Beaucoup de ces activités sont vraisemblablement conduites par des vers autonomes en train de tenter de se reproduire en se propageant. D'autres sont dues à des attaques ciblées, d'autres encore à des attaques limitées dans le temps ou l'espace. L'analyse détaillée des différents constituants de telles attaques, tels qu'ils nous sont offerts par SGNET, nous permet d'en savoir plus sur l'apparition, l'évolution, la réutilisatoin de toute ou partie des briques de bases nécessaires à mener à bien des attaques. Dans ce qui suit, nous expliquons sur base de quelques exemples concrets, les pistes que nous avons commencé à explorer.

SGNET, comme nous l'avons expliqué, nous permet d'obtenir de l'information sur les différentes phases du modèle epsilon-gamma-pi-mu [17]. Chaque phase d'une attaque de type injection de code est traitée par une entité distincte de notre système distribué. La mise en œuvre de chaque phase livre des informations sur les caractéristiques de l'instance observée. Par exemple, l'interaction réseau impliquée dans la phase ϵ est associable à un chemin particulier suivi dans la machine à états finis Scriptgen. Toutes les informations sur chaque phase collectées dans SGNET sont stockées dans une base de données centralisée. De façon semblable à ce qui avait été fait dans le projet Leurré.com case, cette information est enrichie par différents outils d'analyse. Par exemple, chaque malware collecté par SGNET est soumis au système Anubis [3] afin d'obtenir de l'information sur son comportement lorsqu'il est exécuté. Ainsi, la base de données SGNET nous offre une grande richesse d'informations sur les phases (ϵ), (π) et (μ).[4]

Dans notre modèle, une attaque est identifiée par un tuple (ϵ, π, μ), chaque dimension se voit assigner un identifiant qui représente le *type d'interaction* particulière que cette attaque à eu avec notre plateforme pour la phase concernée. L'étude des relations existantes entre les différentes valeurs dans cet espace à trois dimensions offre une perspective nouvelle sur la façon dont les briques de base d'une attaque sont réutilisées dans des attaques a priori différentes.

L'identification du *type d'interaction* n'est pas toujours une chose facile car nous avons de plus en plus souvent à faire avec des codes polymorphiques et il faut trouver

[4] A ce stade du déploiement, les informations collectées sur l'étape γ n'ont pas relevé d'éléments intéressants et ne figurent donc pas dans cette analyse.

une bonne façon de regrouper ensemble des choses qui, d'aspect extérieur, sont différentes.

Les familles de malware telles que Allaple [15] utilisent des techniques de mutation polymorphique pour générer une nouvelle version du code à chaque tentative de propagation. Une telle façon de procéder permet au binaire d'être différent à chaque mutation, rendant sa détection plus difficile. De notre point de vue, l'utilisation de cette technique nous amène à détecter un très grand nombre de malwares qui semblent différents alors qu'ils sont identiques et cela complique la tâche d'attribution de l'identifiant correct pour la phase μ. La question à résoudre revient à trouver une façon de faire pour être certain que la même valeur de μ va être donnée à deux codes binaires qui semblent totalement différents mais qui ne sont en fait que des variations polymorphiques d'une même souche.

Nous avons choisi de résoudre ce problème en essayant de trouver des invariants existant dans les différentes copies polymorphiques. Nous avons trouvé ces invariants dans le comportement dynamique du code. C'est donc sur base de l'observation du comportement dynamique d'un nombre important de binaires que nous apprenons automatiquement les éléments importants qui nous permettent de classer les codes binaires reçus.

Pour chacune des 29283 attaques d'injection de code observées par SGNET sur une période de 8 mois de janvier à Août 2008, nous avons considéré les caractéristiques présentées dans la Table 7. Pour chaque dimension, nous avons étudié les vecteurs caractéristiques correspondants, découvert les patterns les plus fréquents et utilisé ces patterns pour regrouper les binaires. Dans la suite de ce document, nous ferons références aux termes de e-cluster, p-cluster et m-cluster pour indiquer les groupes d'activités observées selon l'axe ϵ, π ou μ

6.1 Degrés de liberté

Nous avons essayé de prendre le plus de recul possible, dans un premier temps, par rapport à l'analyse des relations entre les trois dimensions (ϵ, π, μ). Pour cela, nous avons compté le nombre de combinaisons différentes identifiées au cours des 8 mois de la période d'observation. Nous avons ainsi énuméré le nombre de clusters, et le nombre de combinaisons de 2 ou 3 valeurs. Dans le cas multi dimensionnel, nous avons comparé le nombre de combinaisons observées avec le nombre théorique maximal afin d'avoir une idée de la variabilité relative constatée.

Tab. 7. Information prise en compte

Exploit	Porte visée
	Identifiant du chemin
	Alertes générées par Snort
Shellcode	Hash du shellcode
	Interaction avec l'émulateur
	Type téléchargement du malware (poussé par l'attaquant ou tiré par la victime)
	Protocole utilisé pour le chargement du malware
	Hôte impliqué dans le chargement (le même que l'attaquant ou un site tiers)
	Porte impliquée dans le chargement
Malware	Hash du binaire
	taille
	Nombre de mutexes créés
	Nom des processus créés
	Nombre de sections déclarées dans l'entête PE
	Version du linker déclarée dans l'entête PE
	Nom du Packer trouvé dans la base PEiD
	Nombre de sections dans l'entête PE marquées comme exécutables et inscriptibles

e-clusters	20
p-clusters	58
m-clusters	74
combinaisons (e,p)	107 (9.22%)
combinaisons (p,m)	186 (4.33%)
combinaisons (e,p,m)	290 (0.33%)

Comme on peut le voir dans la table, la plus grande variabilité est introduite par la combinaison des exploits et du payload : 20 exploits différents ont été combinés avec 58 payload différents de 107 façons différentes, ce qui représente à peu près 9% de toutes les combinaisons possibles.

Ces résultats semblent suggérer que les exploits sont très fréquemment réutilisés dans des variantes différentes d'attaques et combinés avec des versions personalisées du payload. En effet, durant les 8 mois de l'expérience, chaque e-cluster a été combiné, en moyenne, avec 5.9 p-clusters différents et 21.2 m-clusters différents. Le même type de payload a été fréquemment utilisé par différentes familles de malware : en moyenne chaque type de payload a été utilisé pour charger 6.2 m-clusters différents.

6.2 Quelques cas intéressants

Il est intéressant de se pencher plus avant sur certaines régions de l'espace (ϵ, π, μ) afin de mieux comprendre l'impact de cette variabilité sur quelques cas concrets. Dans

ce qui suit, nous portons notre attention sur trois cas intéressants, associés à deux types de vulnérabilités particulières, ainsi qu'aux stratégies de propagation employées par une famille de malware associée à un m-cluster spécifique.

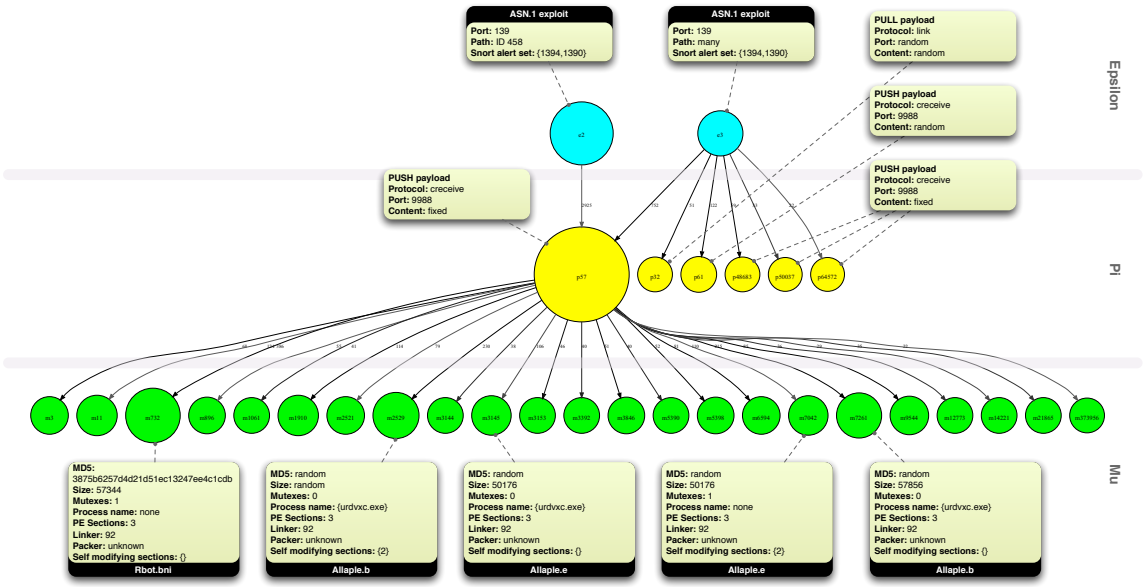


Fig. 16. L' exploit ASN.1 (porte 139)

La vulnérabilité ASN.1 La Figure 16 offre un aperçu de toutes les attaques d'injection de code associées avec l'exploitation de la vulnérabilité ASN.1 (MS04-007) sur la porte TCP 139.

Dans ce cas particulier, il y a une très forte corrélation entre les deux premières dimensions. La totalité des exploits *e2* poussent toujours le même type de payload (*p57*). Ce payload exécute un simple programme qui se lie à la porte TCP 9988 et y reçoit le contenu du malware envoyé par l'attaquant. Un tel comportement est facile à reconnaître à partir des traces réseau et à bloquer : il est rare, en effet, qu'un hôte accepte une connexion entrante sur une telle porte. Bien que cette façon de faire soit très simple et, a priori, facile à empêcher, ce payload est responsable du chargement d'un grand nombre de m-clusters différents. Pour chacun de ces clusters, nous indiquons sur la Figure 16 les informations associées aux malwares, telles que données par l'antivirus Kaspersky, que l'on trouve dans le m-cluster correspondant.

Beaucoup des m-clusters impliqués dans cet exemple sont associés aux différentes variantes du ver polymorphique Allaple. Ces variantes partagent toutes la même stratégie de propagation mais ont des différences au niveau de leur comportement lorsqu'on les exécute sur l'hôte infecté. La même stratégie, c'est à dire combinaison d'étapes (ϵ, π), est également utilisée par d'autres types de malware qui ne sont pas liés à Allaple, tels que le m-cluster 732 qui est associé au bot IRC Rbot.bni.

La famille des malwares Rbot.bni La stratégie de propagation utilisée par le ver Rbot.bni est représentée dans la Figure 17. Il est intéressant de constater que, alors que la plupart des infections qui lui sont associées ont déjà été observées lors de l'analyse précédente dû à l'exploit lié à ASN.1 sur la porte 139, les sondes SGNET ont observé d'autres moyens de propager ce code malveillant. En effet, cette famille de malware peut arriver sur une victime suite à l'utilisation d'un exploit ASN.1 sur la porte 445 or par un exploit DCOM RPC sur la porte 135. Alors que tous les exploits associés à ASN.1 utilisent le payload *p57* décrit précédemment, l'exploit RPC DCOM (*e22*) utilise une stratégie totalement différente. Les exploits liés à cette vulnérabilité obligent la victime à ouvrir une porte et l'attaquant y pousse le code malveillant à partir d'une porte source aléatoire.

La vulnérabilité DCOM RPC Dans ce qui précède, nous avons montré un cas de corrélation importante entre un type d'exploit et un payload. La Figure 18 présente un scénario totalement différent observé pour une autre vulnérabilité, lié à DCOM RPC sur la porte 135. C'est le deuxième vecteur de propagation du malware Rbot.bni dont nous avons déjà parlé. La différence avec la Figure 16 est frappante : dans ce cas-ci, une très grande variabilité existe entre le type d'exploitation et le payload correspondant.

Trois classes différentes de payload peuvent être identifiées : un payload de type *PULL* où la victime vient charger le malware chez l'attaquant, sur la porte 2755 ; un payload de type *PUSH* où la victime attend que l'attaquant lui envoie le malware sur une porte aléatoire à l'aide du protocole *blink* ; enfin, un grand nombre de clusters liés à un payload de type *PULL* à nouveau qui utilise le protocole *link*.

La variabilité en termes de payloads est également constatée par la variabilité en termes du nombre de variantes de malware poussées suite aux différentes combinaisons (ϵ, π). Aucun des clusters μ de la Figure 18 ne correspond à des malwares polymorphiques mais tous sont associés avec des canaux de Commande et Contrôle de type IRC.

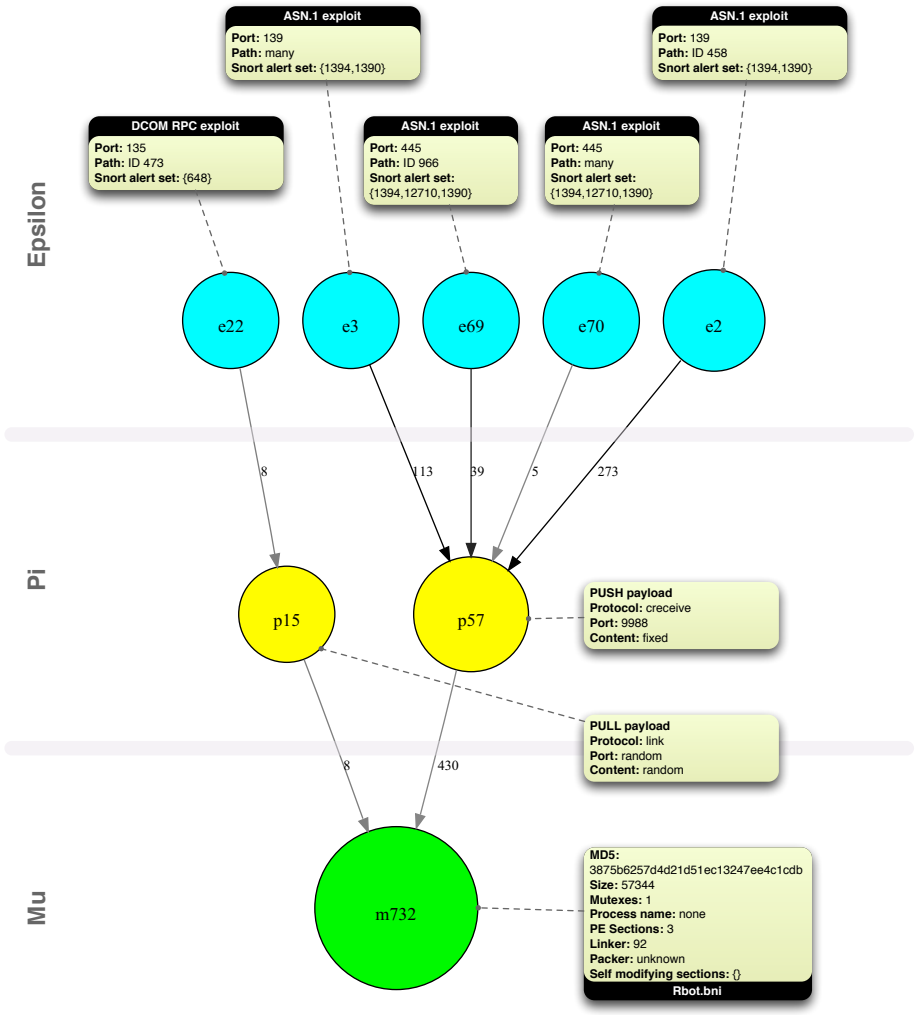


Fig. 17. Capacité de propagation du group μ m732

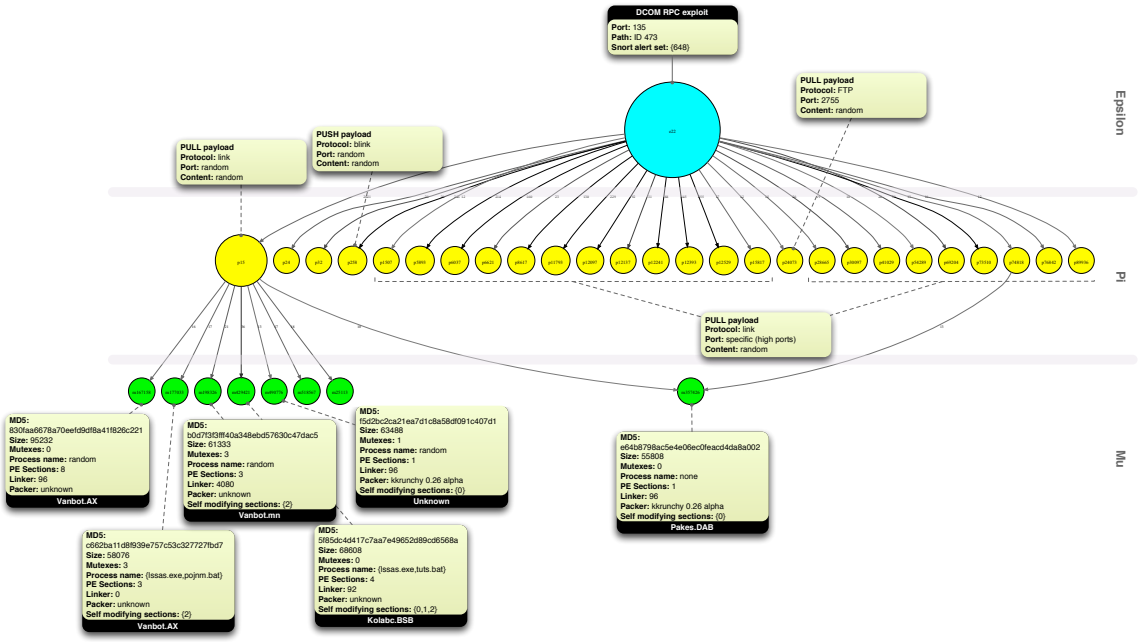


Fig. 18. L'exploit DCOM RPC

7 Conclusions

Dans ce document, nous avons présenté l'environnement du projet Leurré.com et celui de son successeur SGNET développé dans le cadre du projet WOMBAT. Nous avons expliqué à l'aide d'exemples simples la richesse d'information qu'ils offrent aux analystes. De plus, nous avons présenté quelques résultats initiaux obtenus par l'analyse en profondeur de certains évènements d'attaques tels que définis dans la Section 4. Une méthode a été décrite pour leur identification dans des traces réseau obtenues grâce à des pots de miel et pour leur regroupement afin d'inférer l'existence de ce que nous avons appelé les *armées de zombies*. Leurs caractéristiques principales ont été données. Dans la Section 5, nous avons mis en exergue l'utilité de l'analyse multi dimensionnelle pour caractériser des groupes d'évènements et pour émettre des hypothèses sur leur cause originelle. A l'aide d'exemples, nous avons montré comment extraire ce type de connaissance des traces de pots de miel. Enfin, dans la Section 6, en tirant parti de la plus grande richesse d'information à notre disposition grâce au système SGNET, nous avons démontré l'existence de relations subtiles entre des malwares au niveau des tuples (ϵ, π, μ) correspondant.

8 Remerciements

Ce travail a été partiellement financé par la Commission Européenne à travers le projet FP7-ICT-216026-WOMBAT du 7ème programme cadre. Les opinions exprimées dans ce document sont celles des auteurs et ne reflètent pas nécessairement les vues de la Commission Européenne.

Références

1. ALMODE Security. Home page of disco at at http ://www.altmode.com/disco/.
2. P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The Nepenthes Platform : An Efficient Approach to Collect Malware. *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, September 2006.
3. U. Bayer, C. Kruegel, and E. Kirda. *TTAnalyze : A Tool for Analyzing Malware*. PhD thesis, Master's Thesis, Technical University of Vienna, 2005.
4. I. Bomze, M. Budinich, P. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of Combinatorial Optimization*, volume 4. Kluwer Academic Publishers, Boston, MA, 1999.
5. F. M. C. R. Center. Web security trends report q1/2008, http ://www.finjan.com/content.aspx ?id=827, sep 2008.
6. CERT. Advisory CA-2003-20 W32/Blaster worm, August 2003.
7. Z. Chen, L. Gao, and K. Kwiat. Modeling the spread of active worms. In *Proceedings of IEEE INFOCOM*, 2003.
8. M. P. Collins, T. J. Shimeall, S. Faber, J. Janies, R. Weaver, M. D. Shon, and J. Kadane. Using uncleanliness to predict future botnet addresses. In *IMC '07 : Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 93–104, New York, NY, USA, 2007. ACM.

9. E. Cooke, M. Bailey, Z. M. Mao, D. Watson, F. Jahanian, and D. McPherson. Toward understanding distributed blackhole placement. In *WORM '04 : Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 54–64, New York, NY, USA, 2004. ACM Press.
10. J. Crandall, S. Wu, and F. Chong. Experiences using Minos as a tool for capturing and analyzing novel worms for unknown vulnerabilities. *Proceedings of GI SIG SIDAR Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, 2005.
11. M. Dacier, F. Pouget, and H. Debar. Attack processes found on the internet. In *NATO Symposium IST-041/RSY-013*, Toulouse, France, April 2004.
12. M. Dacier, F. Pouget, and H. Debar. Honeypots, a practical mean to validate malicious fault assumptions. In *Proceedings of the 10th Pacific Ream Dependable Computing Conference (PRDC04)*, Tahiti, February 2004.
13. M. Dacier, F. Pouget, and H. Debar. Leurre.com : On the advantages of deploying a large scale distributed honeypot platform. In *Proceedings of the E-Crime and Computer Conference 2005 (ECCE'05)*, Monaco, March 2005.
14. DShield. Distributed Intrusion Detection System, www.dshield.org, 2007.
15. F-Secure. Malware information pages : Allaple.a, http ://www.f-secure.com/v-descs/allaplea.shtml, December 2006.
16. A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series, 1988.
17. C. Leita and M. Dacier. Sgnet : a worldwide deployable framework to support the analysis of malware threat models. In *Proceedings of the 7th European Dependable Computing Conference (EDCC 2008)*, May 2008.
18. C. Leita and M. Dacier. SGNET : Implementation Insights. In *IEEE/IFIP Network Operations and Management Symposium*, April 2008.
19. C. Leita, M. Dacier, and F. Massicotte. Automatic handling of protocol dependencies and reaction to 0-day attacks with ScriptGen based honeypots. In *RAID 2006, 9th International Symposium on Recent Advances in Intrusion Detection, September 20-22, 2006, Hamburg, Germany - Also published as Lecture Notes in Computer Science Volume 4219/2006*, Sep 2006.
20. C. Leita, K. Mermoud, and M. Dacier. Scriptgen : an automated script generation tool for honeyd. In *Proceedings of the 21st Annual Computer Security Applications Conference*, December 2005.
21. C. Leita, V. Pham, . Thonnard, E. Ramirez-Silva, F. Pouget, E. Kirda, and M. Dacier. The Leurre.com Project : Collecting Internet Threats Information using a Worldwide Distributed Honeynet. In *1st WOMBAT open workshop*, April 2008.
22. Maxmind Product. Home page ot the maxmind company at http ://www.maxmind.com.
23. D. Moore, C. Shannon, G. Voelker, and S. Savage. Network telescopes : Technical report. *CAIDA, April*, 2004.
24. S. Needleman and C. Wunsch. *A general method applicable to the search for similarities in the amino acid sequence of two proteins*. J Mol Biol. 48(3) :443-53, 1970.
25. Netgeo Product. Home page of the netgeo company at http ://www.netgeo.com/.
26. V.-H. Pham and M. Dacier. Honeypot traces forensics : The observation view point matters. Technical report, EURECOM, 2009.
27. V.-H. Pham, M. Dacier, G. Urvoy Keller, and T. En Najjary. The quest for multi-headed worms. In *DIMVA 2008, 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, July 10-11th, 2008, Paris, France*, Jul 2008.
28. G. Portokalidis, A. Slowinska, and H. Bos. Argos : an emulator for fingerprinting zero-day attacks. *Proc. ACM SIGOPS EUROSYS*, 2006.
29. F. Pouget, M. Dacier, and V. H. Pham. Understanding threats : a prerequisite to enhance survivability of computing systems. In *IISW'04, International Infrastructure Survivability Workshop 2004, in conjunction with the 25th IEEE International Real-Time Systems Symposium (RTSS 04) December 5-8, 2004 Lisbonne, Portugal*, Dec 2004.
30. T. C. D. Project. http ://www.cymru.com/darknet/.
31. N. Provos. A virtual honeypot framework. In *Proceedings of the 12th USENIX Security Symposium*, pages 1–14, August 2004.
32. M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *ACM SIGCOMM/USENIX Internet Measurement Conference*, October 2006.

33. E. Ramirez-Silva and M. Dacier. Empirical study of the impact of metasploit-related attacks in 4 years of attack traces. In *12th Annual Asian Computing Conference focusing on computer and network security (ASIAN07)*, December 2007.
34. J. Riordan, D. Zamboni, and Y. Duponchel. Building and deploying billy goat, a worm detection system. In *Proceedings of the 18th Annual FIRST Conference*, 2006.
35. I. M. Sensor. http ://ims.eecs.umich.edu/.
36. TCPDUMP Project. Home page of the tcpdump project at http ://www.tcpdump.org/.
37. The Metasploit Project. www.metasploit.org, 2007.
38. O. Thonnard and M. Dacier. A framework for attack patterns' discovery in honeynet data. *DFRWS 2008, 8th Digital Forensics Research Conference, August 11- 13, 2008, Baltimore, USA*, 2008.
39. O. Thonnard and M. Dacier. Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology. In *ICDM'08, 8th IEEE International Conference on Data Mining series, December 15-19, 2008, Pisa, Italy*, Dec 2008.
40. L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9 :2579–2605, November 2008.
41. T. Werner. Honeytrap. http ://honeytrap.mwcollect.org/.
42. M. Zalewski. Home page of p0f at http ://lcamtuf.coredump.cx/p0f.shtml.