École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

Thèse

présentée pour obtenir le grade de docteur

de l'Ecole Nationale Supérieure des Télécommunications

Spécialité : Informatique et Réseaux

Van-Hau PHAM

De l'identification d'événements d'attaques dans des traces recueillies sur des pots de miel

Soutenue le 28 septembre 2009 devant le jury composé de

École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

PhD Thesis

Ecole Nationale Supérieure des Télécommunications

Communications and Electronics department

Computer Science group

Van-Hau PHAM

Honeypot traces forensics by means of attack event identification

Defense date 09, 28 2009

Acknowledgements

First of all, I would like to express my deepest gratitude to my advisor Prof. Dr. Marc Dacier who has accepted me in his team for almost 6 years. I first started with him for a 6 months internship, then continued as a software engineer and eventually ended up as a PhD student. During this long period, he has always shown his unbelievable patience while encouraging me and giving me time to learn. His constant availability to understand the problem I had, to give me instructions to correct my errors, this enabled me to move faster towards the final goal. Without him, I can not imagine how I could have finished this thesis. The endless lessons I have learned from him will definitely help me in my future professional and personal life. Now looking back, I just realize that this is one of the best opportunities that I have ever had in my life. For the lack of better words, thanks Marc to make it happened.

I would also like to thank Dr. Guillaume Urvoy-Keller and Dr. Taoufik En-Najjary for all the discussions we have had together. I always found practical answers and kindness from Guillaume and rigorous mathematical solutions from Taoufik. They both have contributed at numerous occasions to the success of my thesis.

Also, it was a privilege for me to work with Corrado Leita and Olivier Thonnard. Besides Marc, they were the people that I have mostly interacted with during my thesis work. Their input and ideas are present here and there in my final thesis document.

I am also grateful to Dr. Fabien Pouget with whom I have spent almost my first two years. He has continuously helped me from the very first days in France and he was also, unsurprisingly, the one who helped me getting used to my new living environment. Taking advantage of this occasion, I would like to also thank Thierry Woelfflé, who has helped me in solving the daily life problems during my very first days in this lovely country.

I would also like to dedicate my gratitude to the members of the jury for their comments and the time spent traveling to be present for my graduation day.

I would also like to thank my colleagues at Eurecom, the endless help from the IT department. I'm especially grateful to Valérie Loisel, Christophe Lonziano, Patrick Petitmengin and Pascal Mayani. Also, I have always been very well received and helped by the members of the administration, namely Gwenaelle Le Stir, Sophie Salmon and all members in charge of the secretary throughout these years. Thanks to them, I have been able to focus solely on my work.

One of the factors that made me happy and relaxed during the, sometimes, difficult days was the presence of my Vietnamese friends in Sophia Antipolis. I can not remember how much time we have shared, how many jokes they have made... This contributed in maintaining a good balance between my social and professional life, hereby leading to an increase productivity in my work. So, thanks a lot. A special thanks goes to Lan for all the support and encouragement she has offered me during my thesis. I want to emphasize how important this has been for me.

And last but not least, this thesis is dedicated to my parents, brother, and sisters. The ones I know I can turn to whenever I need them. Thanks for the endless support you have provided me during all these years.

Résumé

La sécurité de l'Internet représente un souci majeur de nos jours. De nombreuses initiatives sont menées qui cherchent à offrir une meilleure compréhension des menaces. Récemment, un nouveau domaine de recherches a vu le jour qui vise à étudier le mode opératoire des attaquants et à identifier les caractéristiques des groupes responsables des attaques observées. Le travail réalisé dans cette thèse concourt à cet objectif. Nous montrons que, à partir de traces réseau obtenues à partir d'un réseau mondial de pots de miel sur une période de deux ans, il est possible d'extraire de la connaissance significative et utile sur les attaquants. Pour atteindre ce but, la thèse offre plusieurs contributions importantes. Tout d'abord, nous montrons que les traces d'attaques peuvent être groupées en trois classes distinctes, correspondant à des phénomènes d'attaque différents. Nous avons défini, implémenté et validé des algorithmes qui permettent de classifier un très grand nombre de traces selon ces trois catégories. Deuxièmement, nous montrons que, pour deux de ces classes, il est possible d'identifier des micro et macro événements d'attaques présents durant un nombre limité de jours. Ces événements sont des éléments importants pour identifier des activités spécifiques qui, autrement, auraient été perdues dans le bruit diffus des autres attaques. Ici encore, un environnement été défini, réalisé et validé à l'aide de deux ans de trace. Des centaines d'événements ont été ainsi trouvés dans nos traces. Enfin, nous montrons que, en regroupant ces événements, il es possible de mettre en lumière le mode opératoire des organisations responsables des attaques. La validation expérimentale de notre approche nous a menés à l'identification de dizaines de ce que nous appelons des armées de zombies. Leurs caractéristiques principales sont présentées dans la thèse et elles révèlent des informations importantes sur la dynamique associée aux attaques observables sur l'Internet.

Abstract

Internet security is a major issue nowadays. Several research initiatives have been carried out to understand the Internet security threats. Recently, a domain has emerged called attack attribution that aims at studying the modus operandi of the attacks and at identifying the characteristics of the groups responsible for the observed attacks. The work presented in this thesis participates to the efforts in this area. We show in this work that, starting from network traces collected over two years on a distributed system of low interaction honeypots, one can extract meaningful and useful knowledge about the attackers. To reach this goal, the thesis makes several important contributions. First of all, we show that attack traces can be automatically grouped into three distinct classes, corresponding to different attack phenomena. We have defined, implemented and validated algorithms to automatically group large amount of traces per category. Secondly, we show that, for two of these classes, so called micro and macro attack events can be identified that span a limited amount of time. These attack events represent a key element to help identifying specific activities that would, otherwise, be lost in the so called attack background radiation noise. Here too, a new framework has been defined, implemented and validated over 2 years of traces. Hundreds of significant attack events have been found in our traces. Last but not least, we showed that, by grouping attack events together, it was possible to highlight the modus operandi of the organizations responsible for the attacks. The experimental validation of our approach led to the identification of dozens of so called zombie armies. Their main characteristics are presented in the thesis and they reveal new insights on the dynamics of the attacks carried out over the Internet.

Table of contents

Synthèse en français

Motivation

Avoir une perception précise du paysage des menaces sur Internet, et la capacité à déterminer l'identité et la position des attaquants est le centre de préoccupation d'une partie de la communauté des chercheurs en sécurité. Ces sujets sont généralement qualifiés de compréhension de la situation et de l'attribution d'attaques [24]. Les défis auxquels nous sommes confrontés en traitant ces problèmes sont que les attaquants essaient de masquer leur identité. Par conséquent, il est difficile d'identifier les responsables. Ceci est particulièrement vrai à la lumière de l'innovation sans fin des outils d'attaques. A titre d'exemple, durant les six derniers mois de 2008, Symantec a découvert 1,656,227 nouveaux codes malveillants [120]. Ce processus est avant tout motivé par le profit commercial des activités de cybercriminalité [128]. Plusieurs modèles ont été examinés. En fait, il existe de nombreuses façons d'utiliser des machines compromises, par exemple, en volant des informations sensibles, en effectuant la fraude aux clics [52, 51], en vendant des machines zombies [105, 39], en envoyant du spam [6, 101, 126, 114, 53, 13], ou encore en faisant des attaques du type DDoS [54, 107, 72, 84]. En conséquence, Internet est devenu un endroit attrayant pour de nombreux groupes d'attaquants avec des diversités en stratégies d'attaque, compétences et objectifs [40].

Entre autres, les menaces trouvées sur Internet ont les deux caractéristiques suivantes :

– **Les activités malveillantes sont localisées**. Il a été démontré que les attaques observées sur Internet ne sont pas uniformes ni en type, ni en intensité selon endroit où elles sont observées [10, 22, 18, 28, 57, 81, 92, 91]. Par exemple, dans [18] Pouget et al a fourni une étude comparative des attaques observées sur deux capteurs identiques, un en France et l'autre à Taiwan. L'étude a montré des différences entre les deux traces d'attaques par rapport à plusieurs aspects tels que les services les plus attaqués, les domaines des attaquants, les patterns d'attaques spécifiques par réseau. La localité des attaques est également confirmée par d'autres travaux [83, 22, 57]. Les vers Nimda [37], CodeRedII [75], Blaster [11, 15] sont célèbres pour leur mécanisme de propagation spécifique. Par ailleurs, en profitant de l'information dans la table de routage BGP, un "routing worm" peut non seulement se propager plus vite, mais peut également choisir les cibles spécifiques tels qu'un pays, une entreprise, une ISP, une AS [139].

– **Les machines compromises, organisés en réseaux de zombies, sont de plus en plus contrôlables**. Un réseau de zombies est un réseau de machines compromises, appelés bots, qui sont sous le contrôle de leur botmaster. Bien que les bots peuvent partager plusieurs caractéristiques avec d'autres classes de malwares, leur caractéristique discriminante est d'utiliser une *canal de contrôle et commande (command and controle channel)* pour recevoir des commandes de leur maître. Grâce à cette propriété, les attaquants peuvent exploiter les machines affectées de manières différentes à objectifs différents. Les réseaux de zombies ont été vus se déplaçant à partir d'une infrastructure centralisée [12, 40, 51, 136] vers une distribuée [44, 50, 116, 115, 113, 127], en utilisant d'abord des protocoles faciles à détecter [12, 40, 51, 136] aux plus difficiles à détecter [20, 30, 51, 114]. Selon [100], *27 % de toutes les tentatives observés à partir de notre Darknet distribués peuvent être directement attribués aux activités de propagation de botnets.*

Ces tendances soulignent le fait que les dangers d'une attaque ne résident pas seulement dans son exploit, c'est à dire dans les mesures prises afin de compromettre une machine, mais aussi dans la façon dont elle est utilisée (ou le mode opératoire de l'outil). Par exemple, la localisation rend les outils d'attaques plus furtifs et la contrôlabilité donne à l'attaquant la possibilité d'avoir différentes options sur la façon d'exploiter les machines infectées. Alors que les modes opératoires sont importants, ils ne peuvent pas toujours être tiré ni par l'observation de l'attaque individuellement, ni par l'analyse de ses binaires. Ceci est peut-être dû à plusieurs raisons. Par example, le mode opératoire d'un outil n'est pas toujours visible en une seule attaque. Par exemple, en étudiant individuellement l'attaque entre une source et une destination, il est impossible de savoir s'il s'agit d'un scan du type balayage. En outre, des informations sur le mode opératoire ne sont pas toujours présentes dans les outils. Par exemple, dans le cas d'un réseau de zombies, les cibles des attaques sont reçues du botmaster, elles ne sont pas définies par les bots. Par conséquent, nous devons examiner les sources d'attaques dans leur contexte pour comprendre vraiment les processus d'attaques derrière eux, les stratégies et les motivations des attaquants.

Positionnement de travail

Pour évaluer le niveau de menace sur Internet, les chercheurs ont pris plusieurs initiatives telles que la collecte des échantillons de malwares [65, 8], l'analyse de la dynamique des malwares [63, 26, 108], l'évaluation des URL [99]. La collecte des traces d'attaque devient une approache couramment acceptée dans la communauté de sécurité réseaux [32, 77, 9, 27, 14, 36, 81, 96, 76]. Grâce aux données recueillies, les analystes appliquent des modèles mathématiques pour déduire les caractéristiques des menaces actuelles [2, 3]. Pour que cette approche fonctionne, nous devons posséder, entre autres choses, un ensemble de données propre et représentatif. Dans le cadre de cette thèse, nous voulons étudier le problème de sécurité sur Internet grâce à l'utilisation de traces des attaques. Pour pouvoir avoir de bons résultats, il faut un ensemble de données propre comme input pour notre approche. Dans notre cas, pour permettre d'évaluer la nature des menaces sur Internet, nous devons recueillir des traces des attaques de différents endroits dans le monde. Les traces des attaques générées par un outil d'attaque peuvent dépendre de l'environnement avec lequel il interagit. Ainsi, pour pouvoir comparer des traces, elles doivent être obtenues grâce au même type de capteur. Afin de créer un ensemble de données propres, depuis 2003, EURECOM a déployé des pots de miel pour observer et recueillir des traces d'attaque à différents endroits sur Internet. *Un pot de miel est un système d'information dont la valeur réside dans l'utilisation non autorisée ou illicite de cette ressource* [109]. L'ensemble de données recueilli à long terme grâce à cette infrastructure nous offre la possibilité d'observer l'évolution des menaces sur une longue période de temps. Du point de vue des réseaux attaqués, nous ne disposons pas d'informations sur la façon dont les attaques sont effectivement lancées. Ceci rend difficile l'explication des traces des attaques que nous observons. En fait, des traces observées peuvent être la combinaison des interactions de plusieurs activités, exploitées par des attaquants différents. Il est habituellement difficile de dire si deux attaques sont liées ou non. En conséquence, il est difficile d'expliquer les phénomènes d'attaques où plusieurs sources d'attaques sont impliquées. Dans l'état actuel du domaine d'analyse de traces d'attaques, la plupart des efforts ont été consacrés à l'évaluation des types d'attaques [88, 81] ou à la compréhension de propriétés de la sécurité d'Internet et moins sur la compréhension de la façon dont les attaques se produisent [69]. Profitant de notre infrastructure distribuée de pots de miel, notre intention est de regrouper les sources d'attaques dont nous supposons qu'elles ont la même cause originalle et ensuite d'étudier les caractéristiques de ces groupes pour déduire les caractéristiques de ces causes originalles. Il est important de souligner que par causes originalles nous entendons non seulement le type de l'attaque, mais également son utilisation particulière par un attaquant spécifique. Pour détecter de tels groupes de machines d'attaques, nous faisons l'hypothèse clé suivante. Les sources d'attaques partagant la même cause auront des distributions particulières communes à la fois en terme de temps et d'espace. Des exemples de tels groupes de sources d'attaques peuvent être, par exemple, dus à des attaques lancées par des ordinateurs dans un réseau de zombies qui attaquaint un sous-ensemble de nos capteurs pendant une période de temps donnée. Les machines, au cours de cette période, sont membres de ce que nous appelons une micro attaque. Plus formellement :

Définition 1 ***Une micro attaque*** *μ est définie par un tuple $(\mathscr{T}, \mathscr{F})$ où $\mathscr{T}$ représente une période de temps limitée, généralement de quelques jours, et $\mathscr{F}$ représente l'empreinte d'une attaque comme on le voit sur nos capteurs. μ est un ensemble d'adresses IP observées dans $\mathscr{T}$ et qui ont été vues laisser les empreintes $\mathscr{F}$ d'un point de vue donné (par exemple un de nos capteurs)*

Premier objectif de recherche : *Dans cette thèse, nous voulons construire un mécanisme automatisé qui peut reconnaître et caractériser les micros attaques existantes dans des traces d'attaques recueillies auprès d'un ensemble distribué de pots de miel similaires*

Notre hypothèse est que certaines micro attaques peuvent être reliées entre elles pour faire un phénomène d'attaques plus vaste qui est appelée une macro attaque.

Définition 2 ***Une macro attaque*** *est un ensemble de micro attaques observées au cours de la même période de temps et au cours de laquelle les séries temporelles correspondantes sont fortement similaires.*

Deuxième objectif de recherche : *Dans cette thèse, nous voulons construire un mécanisme automatisé qui peut reconnaître et caractériser les macro attaques existantes dans des traces d'attaques recueillies auprès d'un ensemble distribué de pots de miel similaires.*

Approche et défis

Pour reconnaître les micro et macro attaques, nous sommes confrontés aux problèmes suivants :

– **Identification de micro attaques :** Des processus d'attaques différents peuvent laisser des empreintes différentes. Par exemple, on pourrait attendre que le trafic généré par un réseau de zombies soit différent à celui généré par un ver. Notre capacité de reconnaissance pourrait également être impactée par la façon dont les traces se regroupent. Par exemple, si un ver vient de peu d'endroits spécifiques sur Internet, et attaque partout dans le monde, il sera facile d'identifier les traces qu'il laisse en les regroupant par l'origine des sources des attaques. Au contraire, si un réseau de zombies est constitué des bots situés partout dans le monde, mais qu'il attaque quelques blocs d'adresses IP spécifiques, il est possible de mettre en évidence un tel comportement spécifique en regroupant les traces par la destination plutôt que par la source.
– **Complexité :** Comme nous le montrerons plus loin, pour détecter les micro et macro attaques, nous avons besoin de comparer les traces des attaques les unes avec les autres. Ceci est une tâche très couteuse pour un grand ensemble de données. Les facteurs qui constituent ce coût sont :
  – Le volume des traces des attaques.
  – La durée de la période pendant laquelle nous recueillons des traces des attaques.
– **Précision :** Il existe de nombreux types de mesures de similarité/distance telles que SAX [70], facteur de corrélation Pearson, Minkowski. Elles ont été inventées dans des contextes historiques différents pour des buts différents. Nous avons besoin d'identifier celle qui convient à notre but, à savoir corréler les traces des attaques observées sur Internet.

Contribution

Nos contributions dans cette thèse sont les suivantes :

- Nous montrons que, en analysant les macro attaques, nous pouvons en apprendre plus sur les outils d'attaques, les modes opératoires de certaines classes d'attaques, ainsi que plusieurs autres caractéristiques des processus d'attaques sur Internet. Il est important de souligner le fait que les macro attaques générées en tant que résultat de notre approche peuvent être et, en réalité, ont été utilisées par d'autres chercheurs dans le contexte de leur propre travail [122, 123].
- Nous démontrons qu'il est possible d'automatiser la détection de macro attaques pour un grand ensemble de données. Nous dévelopons trois solutions pour l'identification de macro attaques qui peuvent s'utiliser dans des contextes différents. Les trois solutions ont été validées expérimentalement.
- Nous montrons que les outils d'attaques peuvent être classés en trois familles selon leur niveau d'activité. La première famille se compose d'outils qui sont utilisées de façon constante. La deuxième famille se compose d'outils qui sont lancés de temps à autre sur une période de quelques jours. La dernière famille se compose d'outils qui sont rarement utilisés plus d'une fois et toujours pendant un ou deux jours seulement. Cette découverte est fondamentale pour réduire le coût de calcul lors de la détection d'événements d'attaques.
- Nous montrons aussi l'impact qu'à la façon dont on groupe les traces des attaques sur notre capacité à détecter certaines micro et macro attaques. Selon que nous utilisons l'origine ou la destination de ces attaques, nous pouvons identifier des macro attaques différents.

Terminologie

Définition 3 ***Une Plateforme*** *est une machine physique qui simule, grâce au programme honeyd [98], la présence de trois machines distinctes. La plateforme est reliée directement à Internet et recueille les traces tcpdump qui sont fournies quotidiennement à la base de données centralisée du projet Leurré.com.*

Définition 4 ***Leurré.com :*** *Le projet Leurré.com est un système distribué de telles plateformes déployées dans plus de 50 endroits dans 30 pays différents (voir [67] pour les détails)*

Définition 5 ***Une Source*** *correspond à une addresse IP observée sur une ou plusieurs plateformes, et pour laquelle le temps d'arrivée entre deux paquets consécutifs reçue reste inférieur à un certain seuil (25 heures).*

Définition 6 ***Une Large Session*** *est l'ensemble de paquets échangés entre une source et une plateforme particulière.*

Définition 7 ***Un Cluster*** *est un ensemble de larges sessions similaires, qui sont, en première approximation, générées par un même outil d'attaque*

Définition 8 ***Une Série Temporelle*** *$\Phi_{T,c,op}$ est une fonction définie sur une période de temps T, T étant définie comme un intervalle de temps (en jours). Cette fonction retourne le nombre de sources par jour, associées au cluster c, observées à partir du point de vue op donné. Le point de vue op peut être soit une plateforme spécifique ou soit un pays d'origine spécifique. Dans le premier cas, $\Phi_{T,c,platform_X}$ retourne, par jour, le nombre de sources appartenant au cluster c qui ont été observées par $platform_X$. De même, dans le second cas, $\Phi_{T,c,country_X}$ retourne, par jour, le nombre de sources appartenant au cluster c qui sont géographiquement situées dans $country_X$. De toute évidence, nous avons toujours : $\sum^{\forall i \in pays} \Phi_{T,c,i} = \sum^{\forall x \in plateformes} \Phi_{T,c,x}$*

Définition 9 ***Une Série Temporelle Rassemblée*** *$\Phi_{T,op}$ est une fonction définie sur une période de temps T, T étant définie comme un intervalle de temps (en jours). Cette fonction retourne le nombre de sources par jour pour un point de vue op. Le point de vue peut être une plateforme spécifique ou un pays d'origine spécifique. En conséquence, la série temporelle rassemblée $\Phi_{T,op}$ est la somme de toutes les séries temporelles de forme $\Phi_{T,*,op}$ ou $\Phi_{T,op} = \sum^{\forall c \in clusters} \Phi_{T,c,op}$*

Détection de micro et macro attaques

Comme une attaque peut se passer à plusieurs endroits, il est nécessaire de corréler des attaques pour l'identifier. C'est pour cela que nous commençions par l'identification de mesure de correlation qui répond bien à notre besoin. La deuxième section présente l'approache pour identifier des événements d'attaques tous en prenant en compte le problème de complexité. La troisième section propose deux usages spéciaux des techniques discutées à la deuxième section pour réduire encore plus le coût de calcul.

Mesures de correlation

Nous avons à ce jour plusieurs techniques qui peuvent être utilisées pour calculer la distance entre deux séries temporelles. Avant de décider laquelle convient le mieux à nos besoins, nous étudions en détail notre contexte d'application. Premièrement, s'agissant de l'exigence opérationnelle, nous avons besoin d'une technique qui est applicable à un ensemble de données de grande taille. Deuxièmement, concernant l'exigence fonctionnelle, dans le cadre des traces des attaques, si les séries temporelles sont stables, nous considérons qu'il n'y a aucune activité spéciale qui se passe. Sinon, si les séries temporelles contiennent de variations ou des pics d'activités, nous considérons que quelque chose de différent se passe. Quand un tel cas se produit, c'est appel à regarder et à identifier des phénomènes d'attaques.

En ce qui concerne le contexte d'application tel que décrit ci-dessus, nous identifions les propriétés obligatoires suivantes pour la fonction de similitude idéal $\mathscr{P}()$

- **Synchronisation :** *$\mathscr{P}()$ ne devrait pas retourner vrai si les pics d'activités ne sont pas synchronisés.* La raison en est que les activités importantes sont les plus intéressantes dans les séries temporelles. Si $\mathscr{P}()$ conclut que les deux

séries temporelles sont corrélées, leurs activités importantes doivent être synchronisées.
- **Evolutivité :** *$\mathscr{P}()$ doit être applicable à un grand ensemble de données.*

En outre, du point de vue pratique, nous proposons deux autres exigences supplémentaires. Ce ne sont pas des exigences obligatoires susceptibles d'exclure une technique, mais elles ont une influence certaine sur le choix final.
- **Détermination de seuils :** Une des tâches de base pour comparer des objets est de déterminer le seuil à partir duquel deux objets sont considéres comme étant, ou non, similaires. Dans le même ordre d'esprit, deux séries temporelles seront, ou non, corrélées sur la base d'une telle valeur. Pour certaines fonctions de corrélation, nous avons des seuils admis couramment, pour d'autres, le seuil doit être fixé manuellement. Par exemple, la fonction de distance SAX est influencée par la taille de l'alphabet, et le ratio de compression. Pour simplifier les choses, *nous plaidons en faveur de fonctions de corrélation $\mathscr{P}()$ ayant des valeurs seuils prédéfinies et largement acceptés.*
- **Besoin de traitement :** Certaines fonctions de corrélation exigent un prétraitement des données, à savoir, leur normalisation. Nous favorisons les fonctions de corrélation $\mathscr{P}()$ qui ne nécessitent pas de telles tâches pour des raisons de coût.

Approche

Pré-traitement

Le problème de surcharge de calcul vient du fait que nous avons plusieurs séries temporelles. En fait, pour savoir si une série temporelle est corrélée avec d'autres, nous devons comparer cette série à toutes les autres. Il est évident que plus grand est le nombre de séries temporelles, plus cher est l'effort de calcul. A titre indicatif, nous avons environ 400000 séries temporelles. Un tel nombre de séries induit de frais de calcul élevé. Pour faire face à cela, nous observons qu'il existe un moyen facile de mettre les séries temporelles en des classes différentes. Nous pouvons distinguer deux types de corrélation de la façon suitvante :
- **corrélation de type intra-classe :** Il consiste à calculer les corrélations des séries temporellesr appartenant à une même classe.
- **corrélation de type inter-classe :** Il consiste à calculer les corrélations des séries temporelles appartenant à des classes différentes.

Si, par design, les classes que nous avons construites sont telles qu'il ne devrait y avoir aucune corrélation significative entre deux séries temporelles appartenant à deux classes différentes, alors nous consacrons l'effort de calcul à la seule évaluation de la "corrélation du type intra-classe".

Nous observons que les séries temporelles peuvent être classées en des catégories différentes en fonction de leur niveau de variabilité. Nous avons montré que les séries temporelles peuvent être classées dans l'une des trois catégories suivantes :

1. **famille de pics** : les séries temporelles dans cette famille possèdent un pic important d'activitiés pendant une période très courte d'un ou deux jours

et presque aucune activité en dehors de ce pic. Le fait que plusieurs sources envoient des paquets simultanément à une seule plateforme dans une courte période de temps montre bien le caractère hautement coordonné des attaques.

2. **famille stable** : les séries temporelles dans cette famille ont un comportement constant à peu près durant toute leur vie.
3. **famille de variation** : les séries temporelles dans cette famille sont caractérisées par de larges variations d'amplitude et, parfois, sur de longues périodes de temps.

Il y a beaucoup d'avantages avec la classification ci-dessus. Premièrement, nous n'avons pas besoin de calculer la *corrélation inter-classes* car les séries temporelles de la famille stable ne peuvent pas être corrélées ni avec celles de la famille de pics ni avec celles dans la famille de variation. De même, les séries temporelles de la famille de variation ne sont pas susceptibles d'être corrélées avec celles dans la famille de pics. Deuxièmement, puisque le calcul de la corrélation des séries temporelles stables n'a pas beaucoup de sens, nous n'avons pas besoin de calculer les *corrélations intra-classe* pour les séries temporelles appartenant à cette catégorie.

Fenêtre glissante

- **Etape 1 : Corrélation avec la fenêtre glissante** Étant donné deux séries temporelles Φ et Ψ, de longueur T, nous voulons identifier les périodes corrélées $Pi = \{start, stop, \Phi, \Psi\}$ où deux séries temporelles Φ et Ψ évoluent de même façon de *start* à *stop*. Pour obtenir ces intervalles de temps, une approche consiste à calculer les coefficients de corrélation pour tous les intervalles possibles de $[a, b]$ avec $(0 \leq a \leq b \leq T)$, mais ce serait extrêmement coûteux. Pour éviter cela, nous utilisons une corrélation calculée sur une fenêtre glissante. L'idée est que l'on calcule les coefficients de corrélation pour les fenêtres successives (avec une longueur fixe, L) de Φ et Ψ, passant de la gauche vers la droite pour obtenir un vecteur de corrélation C.
- **Étape 2 : Identification des intervalles corrélés de deux séries temporelles**
  Étant donné un seuil δ et un vecteur de corrélation C calculé pour deux séries temporelles Φ et Ψ, notre objectif est de déterminer les intervalles de temps durant lesquels ces deux séries temporelles sont considérées comme étant en corrélation. En d'autres termes, nous essayons d'identifier les périodes corrélées $P_i = \{start_i, stop_i, \Phi, \Psi\}$.
- **Étape 3 : Identification des intervalles corrélés communs** Notre prochain objectif est de regrouper les intervalles corrélés en commun $G = (\{T_1, S_1\}$ $, \{T_2, S_2\}, \cdots \{T_k, S_k\})$. Chaque $G_i = \{T_i, S_i\}$ correspond à un ensemble P de paires corrélées, entre les mêmes jours $T_{i,start}$ et $T_{i,stop}$. S_i contient toutes les séries temporelles dans S_i , ou $S_i = \cup P_{j,ts1}, P_{j,ts2} \forall P_j \in P$
- **Étape 4 : Extraction des macro attaques** Sur la base de l'intervalles corrélés communs, nous avons maintenant besoin de regrouper toutes les séries temporelles qui sont mutuellement corrélées entre elles. Pour ce faire, nous utilisons une représentation graphique des paires de corrélation identifiées dans l'étape

précédente de notre algorithme. Les noeuds dans le graphique représentent les séries temporelles sur des périodes de temps corrélees et deux noeuds sont liés si deux séries temporelles correspondantes sont corrélées dans durant la même période de temps. Ensuite, l'identification des macro attaques revient à trouver les sous-graphes connectées

Usages spéciaux de la fenêtre glissante

Approche Breakdown

Cette approche repose sur l'hypothèse que, les macro attaques $e_i = (T_{i,start}, T_{i,end}, S_i)$ doivent appartenir à une des trois catégories suivantes :

1. Une macro attaque e_i qui n'aurait qu'un seul cluster impliqué. En d'autres termes, si S_i consiste en un ensemble de séries temporelles de la forme $\Phi_{T',c,*}$, une telle macro attaque est appelée **macro attaque distribuée**. L'adjectif *distribué* souligne que cette attaque concerne plusieurs endroits.
2. Une macro attaque e_i qui serait liée à un seul point de vue. Dans ce cas, si S_i consiste en un ensemble de séries temporelles de la forme $\Phi_{T',*,op}$, une telle macro attaque est appelée **macro attaque localisée**. L'adjectif *localisée* souligne que cette attaque vise qu'un seul endroit, mais implique plusieurs clusters.
3. Une macro attaque mixte concerne plus d'un cluster et plus d'un point de vue et est composée de plusieurs macro attaques appartenant aux deux catégories précédentes.

Dans cet esprit, soit M le nombre de séries temporelles $\phi_{T,C,op}$, $C = (C_1, C_2, ...)$ l'ensemble de clusters distincts, et $OP = (op_1, op_2, ...)$ l'ensemble de points de vue distincts, on procède comme suit pour détecter les trois types macro attaques mentionnés ci-dessus :

- **Étape 1, Détection de macro attaques distribuées :** Nous classons M séries temporelles dans $|C|$ ensembles $(D_1, D_2, ..., D_{|C|})$. L'ensemble, D_i, ne contient que les séries temporelles établies à partir du cluster c_i, ou $D_i = \Phi_{T',c_i,op} \forall op \in OP$. Nous appliquons ensuite l'approche de corrélation avec la fenêtre glissante telle que présentée plus tôt pour chaque ensemble de données D_i pour détecter les macro attaques distribuées.
- **Étape 2, Détection de macro attaques localisées :** Nous classons les M séries temporelles en, cette fois, $|OP|$ ensembles $(D'_1, D'_2, ..., D'_{|OP|})$. L'ensemble D'_i ne contient que les séries temporelles établies à partir du point de vue op_i ou $D'_i = \Phi_{t',c,op_i}$ $\forall c \in C$. Nous appliquons également l'approche corrélation avec fenêtre glissante pour détecter les macro attaques localisées dans chaque ensemble de données.
- **Étape 3, Détection de macro attaques mixtes :** Cette étape vise à regrouper les macro attaques identifiées précédemment. Supposons que $E = (E_1, E_2, ..., E_n)$ est l'ensemble de toutes les macro attaques détectées au cours de deux étapes précédentes. Toutes les macro attaques ayant une durée de vie commune (ayant le même T_{start} et T_{stop}) et ayant leurs séries temporelles corré-

lées sont fusionnées. Les macro attaques obtenues sont appelés macro attaques mixtes.

Approche des séries temporelles rassemblées

Cette approche vise à réduire les coûts de calcul en adoptant certaines hypothèses concernant les caractéristiques des macro attaques. En fait, nous ne comparons pas les séries temporelles, mais leur rassemblemement. La question réside maintenant dans la façon d'agréger les traces des attaques. Notre observation est que les attaques (ou du moins certaines catégories d'attaques) ne sont pas uniformément distribuées ni dans leur source ni dans leur destination [101, 69, 10, 22, 81], on fait l'hypothèse que s'il y a des macro attaques (liées à certains outils d'attaque) qui ciblent ou viennent de certains endroits, ces attaques auront un impact sur les traces aggrégées de toutes les attaques à ces endroits. Par exemple, prenons le cas de la destination des attaques. Nos platesformes observent un nombre limité d'attaques par jour. Si, à un moment donné, deux plateformes deviennent la cible d'attaques coordonnées, nous faisons l'hypothèse que ceci influencera significativement les *séries temporelles globales observées* sur ces deux plateformes au cours de cette période. Par conséquent, la méthode identifie les groupes de séries temporelles agrégées corrélées sur des intervalles de temps différents. De toute évidence, si l'intensité de l'attaque n'est pas assez élevée pour impacter suffisament les séries temporelles rassemblées au moins à deux endroits, notre méthode va les manquer. Une fois que nous avons les groupes de séries temporelles rassemblées corrélées, nous recherchons les causes, à savoir les clusters qui sont à l'origine de la similitude des séries temporelles rassemblées dans chaque groupe. Une fois que nous les avons trouvés, nous vérifions qu'ils n'existent pas aussi à d'autres endroits que ceux initialement trouvés. Cela pourrait se produire si l'impact de ces attaques sur les autres séries temporelles rassemblées n'était pas assez fort pour les inclure dans le groupe de séries temporelles rassemblées corrélées.

En faisant ceci, nous réduison fortement l'effort de calcul, puisque le montant de séries temporelles agrégées est beaucoup plus petit que celui de toutes les séries temporelles.

Impact des points de vue

Dataset

Nous avons sélectionné des traces d'attaques de 40 des 50 plateformes sur une période de 800 jours. Durant cette période, aucune d'elles n'a été en panne plus de 10 fois et chacune d'eelles a été opérationnelle en continu pendant au moins 100 jours une fois. Ils ont toutes été en place pendant un minimum de 400 jours au cours de cette période. L'ensemble de données collectées est brièvement décrit dans la Table 1.

TS est composé de 3,477,976 sources		
OVP	pays	plateforme
$\|TS\_L1\|$	231	40
$\|TS\_L1'\|$	85 (94,4% TS)	40 (100% TS)
$\|TS\_L2\|$	436,756	395,712
$\|TS\_L2'\|$	2,420	2,127
sources	2,330,244 (67% of TS)	2,538,922 (73% of TS)
$\|TS\_L2\_P'\|$	1,360	1,046
sources	41,418	47,673
$\|TS\_L2\_S'\|$	1,060	1,081
sources	2,288,826	2,491,249

Table 1 – *TS : toutes les sources observées pendant la période étudiée, OVP : point de vue, TS_L1 : ensemble de séries temporelles rassemblées, TS_L1' : ensemble de séries temporelles rassemblées significatives dans TS_L1, TS_L2 :ensemble de séries temporelles, TS_L2' ensemble de séries temporelles après pré-traitement de TS_L2, TS_L2_P' ensemble de séries temporelles appartenant à la famille des pics dans TS_L2, TS_L2_S' ensemble de séries temporelles appartenant à la famille des variations dans TS_L2*

Table 2 – Résultats sur l'identification de macro attaques

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

No.AEs : nombre de macro attaques

Résultats sur l'identification de macro attaques

En appliquant la technique de corrélation avec la fenêtre glissante aux deux ensembles de séries temporelles présentées dans la Table 1, nous avons obtenu les résultats comme présentés dans Table 2. Ces résultats mettent en évidence le fait que selon la façon dont nous décomposons l'ensemble initial de traces d'attaques, à savoir en le divisant par pays d'origine des attaquants ou par plateforme attaquée, différentes macro attaques apparaissent. Pour évaluer le chevauchement entre les événements d'attaques identifiés *AE-set-I* et *AE-set-II* à partir de différents points de vue, nous utilisons le *ratio de source en commun, csr*, mesuré comme suivant :

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

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

La Figure 1 représente les deux fonctions de distribution cumulative correspondant à cette mesure. Le point (x, y) sur la courbe signifie qu'il existe $y * 100\%$ des macro attaques obtenues grâce à $T_{country}$ (resp. $T_{platform}$) qui ont moins de $x * 100\%$

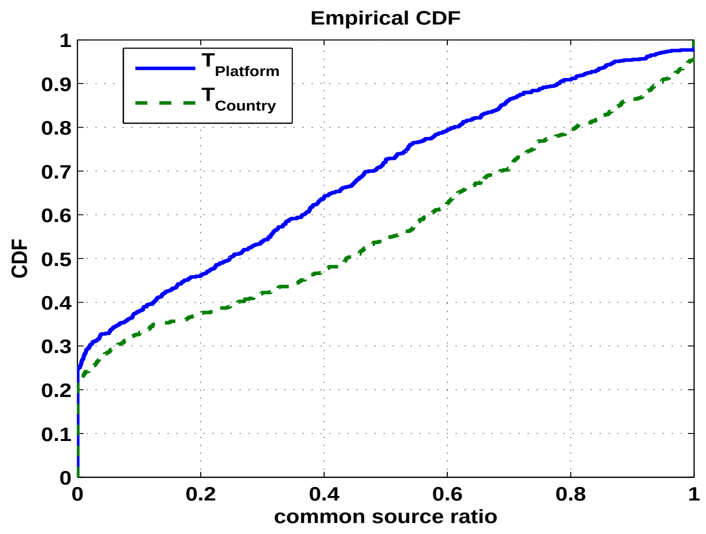


FIGURE 1 – CDF de taux de sources en commun

de sources en commun avec tous les macro attaques obtenues de $T_{platform}$ (resp. $T_{country}$). La courbe $T_{country}$ représente la distribution cumulative obtenue dans le premier cas et la courbe $T_{platform}$ représente le CDF de taux de sources en commun pour $T_{platform}$. Comme nous pouvons le remarquer, environ 23% (resp. 25%) des macro attaques obtenues à partir de $T_{country}$ (resp. $T_{platform}$) n'ont aucune source en commun avec toutes les macro attaques obtenues en utilisant l'ensemble de séries temporelles $T_{platform}$ (resp. $T_{country}$). Cela correspond à 136 (16919 sources) et 171 (75920 sources) macro attaques qui n'ont pas été détectées. Au total, il y a 288825 (resp. 293132) sources présentes dans AE-Set-I (resp. AE-Set-II), mais pas dans AE-Set-II (resp. AE-Set-I). Enfin, il y a au total 867.248 sources ayant participé à l'ensemble macro attaques détectées de deux ensembles à partir de données qui correspondent à 25 % des attaques observées dans la période étudié. Elle montre que les attaques sur Internet ne sont pas totalement aléatoires, mais bien coordonnées de façon stratégique.

Il y a de bonnes raisons qui expliquent pourquoi nous ne pouvons pas compter sur un seul point de vue de pour détecter les événements d'attaques. Elles sont décrites ci-dessous.

– **Division par pays :** Supposons que nous avons un botnet B fait des machines qui sont situées dans l'ensemble des pays $\{X, Y, Z\}$. Supposons que, de temps en temps, ces machines attaquent nos plateformes et laissent des traces qui sont assignées au cluster C. Supposons également que ce cluster C est très *populaire*, c'est à dire que beaucoup d'autres machines dans le monde laissent en permanence des traces sur nos plateformes qui sont assignées à ce cluster. En conséquence, les activités spécifiquement liées au botnet B sont perdues dans le bruit de toutes les autres machines qui laissent de traces appartenant à C. Cela est certainement vrai pour les séries temporelles (tel que définies précédemment) liées à C et cela peut aussi être vrai pour les séries temporelles obtenues en divisant par plateforme,$\Phi_{[0-800),C,platform_i} \forall platform_i \in 1..40$. Toutefois, en divisant les séries temporelles correspondant à cluster C par les pays d'origines des sources, alors il est fort probable que les série temporelles $\Phi_{[0-800),C,country_i} \forall country_i \in \{X, Y, Z\}$ seront fortement corrélées au cours des périodes où le botnet présent dans ces pays seront actives contre nos plateformes. Cela mènera à l'identification d'un ou plusieurs événements d'attaques.

– **Division par platforme :** De même, supposons que nous avons un botnet B' fait de machines situées partout dans le monde. Supposons que, de temps en temps, ces machines attaquent un ensemble spécifique de plateformes $\{X, Y, Z\}$ et laissent de traces qui sont assignés au cluster C. Supposons également que ce cluster C soit très *populaire*, c'est à dire que beaucoup d'autres machines dans le monde laissent des traces en permanence sur toutes nos plateformes qui sont assignées à ce cluster. En conséquence, les activités spécifiquement liées au botnet B' sont perdues dans le bruit de toutes les autres machines laissant de traces appartenant à C. Cela est certainement vrai pour les séries temporelles liées à C et cela peut aussi être vrai pour les séries temporelles obtenues en divisant par pays, $\Phi_{[0-800),C,country_i} \forall country_i \in Big_{pays}$. Toutefois, en divisant les séries temporelles correspondant à cluster C par plateforme, il est alors fort probable que les série temporelles $\Phi_{[0-800),C,platform_i} \forall platform_i \in \{X, Y, Z\}$ seront fortement corrélées au cours des périodes où le botnet attaque nos plateformes. Cela mènera à l'identification d'une ou plusieurs macro attaques.
  L'ensemble de courbes en haut de la Figure 2 représente la macro attaque 79. Dans ce cas, nous voyons que les traces dues au cluster 175309 sont fortement corrélées lorsque nous observons ce groupe par plateforme attaquée. En fait, il y a 9 plateformes impliquées dans cette attaque, ce qui représente un total de 870 sources. Si nous regroupons les mêmes traces par pays d'origine des sources, nous obtenons les courbes du bas de la Figure 2 où le phénomène d'attaque déjà identifié peut être à peine vu.

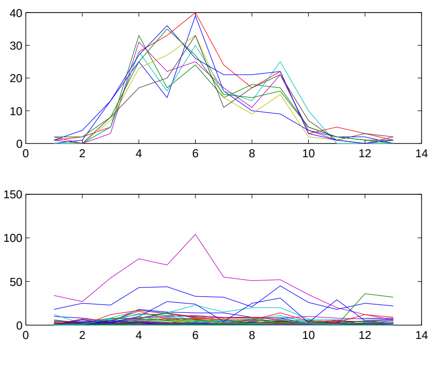


FIGURE 2 – Le graphe du dessus représente l'événement d'attaque 79 lié au cluster 17309 sur 9 plateformes. Celui du bas représente l'évolution de ce cluster par pays. Le bruit des attaques sur d'autres plateformes diminue de manière significative la corrélation des séries temporelles

Identification des armées de zombies

Classification des macro attaques

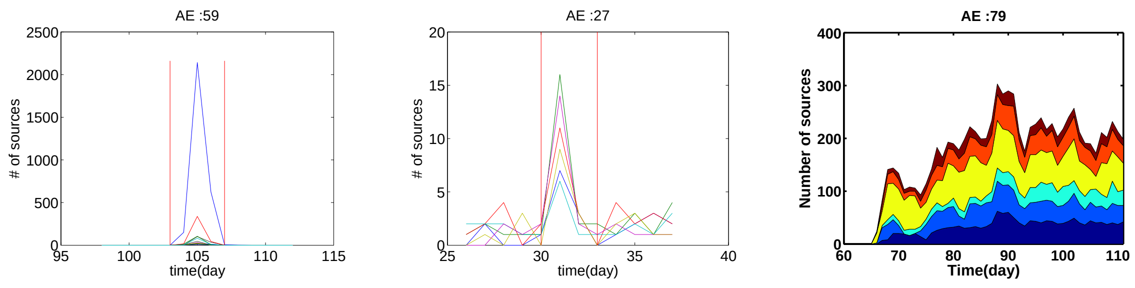
Macro attaques peuvent être générées par des causes différentes. Comme suggéré dans [129, 69], de telles attaques coordonnées peuvent être dues aux activités des vers, des botnets, ou des erreurs de configuration de réseaux. Suivant cette idée, nous classons les macro attaques que nous avons détectées plus tôt en trois classes : *botnet, ver*, et *autres*. Nous nommons la troisième classe *autres* puisque nous ne sommes pas sûrs de la nature de cette catégorie d'activités. Nous observons qu'il existe une classe de macro attaques frappant une seule plateforme, et sur une addresse IP unique et toujours la même sur cette plateforme. En regardant en détail ces attaques, nous observons que les sources ciblent uniquement les numéros de ports élevés, et la plupart d'entre eux sont utilisés par les clients des réseaux P2P. Cela fait à penser en quelque sorte que ces événements d'attaques ne sont pas causés par des vers ni des botnets. En fait, des vers et des botnets peuvent avoir des stratégies différentes pour choisir leurs cibles (hitlist, uniforme, mode aléatoire, ...) mais ils essaient toujours d'augmenter leur efficacité pour se propager en contactant plusieurs machines. Comme notre plateforme se compose de trois adresses IP consécutives, et étant donné une grande quantité de sources, dans chaque événement d'attaques (les tailles moyennes d'un événement d'attaques dans *AE-set-I* et *AE-set-II* sont de 970 et 838 sources, respectivement), nous prévoyons que les macro attaques causés par les botnets et les vers frappent les trois adresses IP quand ils visent une plateforme. Pour cette raison, nous classons les macro attaques frappant une seule adresse IP dans la classe *autres*. Il est important de noter que, les auteurs dans [69] ont également utilisé le nombre limité d'adresses IP contactés afin de classer des attaques dans la classe *erreur de configuration de réseaux*. S'il est facile de différencier les attaques de la classe *autres* de ceux des classes *botnet* et *ver*, Il n'est pas aisé de différencier les activités causées par des vers et celles causées par des botnets. En fait, on s'attendrait à ce que le nombre de source d'attaques laissées par un botnet ait une montée forte et une chute brutale lorque le botnet reçoit des commandes de son maître, tandis que le nombre de sources utilisées par un ver devrait avoir une croissance exponentielle. Toutefois, si un botnet utilise le mode *pull* pour recevoir des commandes, par exemple comme c'est le cas de Phatbot, tous les bots dans ce botnet ne commencent pas à lancer des attaques aux même moment. En fait, comme mentionné dans [131], les Phatbots réveillent toutes les 1000 secondes pour vérifier s'il y a de nouvelles commandes à exécuter. Compte tenu de ce comportement, plutôt que d'une apparition brutale, nous pourrions nous attendre à une montée constante sur un intervalle de 10-20 minutes. Dans notre cas, nous utilisons le pas de temps d'une journée, nous espérons que cela permettra d'éliminer l'artefact du mode *pull*. Donc, dans notre analyse, nous considérons que tous les macro attaques ayant une montée nette et une décroissance forte sont générés par des botnets. Les autres sont attribués à des activités causées par des vers. Le résultat final de la classification de macro attaques est représentée dans la Table 3. Comme nous pouvons le constater, la majorité des événements d'attaques sont causés par des botnets. Il est intéressant

TABLE 3 – Résult sur la classification d'événements d'attaques

	AE-set-I	AE-set-II
Botnet	532	597
Worm	18	36
Others	42	57

de voir qu'il y a plus d'attaques dans la classe *autres* que dans la classe *ver*, et ceci est vrai pour les résultats obtenus par les deux cas *AE-set-I* et *AE-set-II*

Figure 3a représente un exemple un macro attaque de la classe *autres*. Cet attaque se compose de 4.054 sources provenant de plusieurs séries temporelles, ciblant le port 50286/TCP sur une seule adresse IP d'une plateforme située en Chine. Figure 3 b montre le cas d'attaque 27, EA27, dans laquelle le cluster 15238 cible le service Netbios (port 139/TCP) sur les adresses IP de cinq plateformes différentes. En ce qui concerne la forme, les deux attaques AE27 et AE59 se ressemblent : une montée forte et une chute brutale. Et dans les deux cas les attaques subsistent durant une très courte période de temps. Toutefois, AE27 attaque les adresses IP sur un port bien connu et l'AE59 frappe une seule adresse IP sur un port élevé. AE27 est donc classée dans la classe *botnet*. Enfin, la Figure 3c représente attaque 79, AE79, dans laquelle les sources du cluster 65710 attaquent le service Windows Messenger (port 1026/UDP) sur six plateformes différentes situées dans cinq /8 réseaux. La différence avec les deux exemples précédents est que, dans AE79 les attaques sont observable pendant une longue période de temps.



(a)autres (b)botnet (c)ver

FIGURE 3 – a) 4054 sources (appartenant à plusieurs clusters) provenant d'Espagne attaquent une address IP sur le port 50286/TCP b) Cluster 15238 attaque 5 plateformes sur le Service Netbios (port 139/TCP) c) Cluster 65710 envoie paquets contre 6 plateformes sur le service Windows Messenger Popup durant une longue période de temps.

Clustering

Nous croyons qu'il y a des macro attaques qui sont générés par les mêmes cause(s) (ou liées), par exemple le même botnet utilisé pour lancer des attaques différentes à différents points dans le temps. Si nous pouvons trouver des caractéristiques communes à plusieurs attaques, nous pourrions les regrouper et vérifier notre hypothèse. Intuitivement, si deux macro attaques partagent un nombre important de machines d'attaques, il y a de grandes chances qu'elles aient la même cause. Dans cette optique, nous utilisons les adresses IP communes pour mesurer la distance entre les événements d'attaques. Bien sûr, nous ne pensons pas que ce nombre sera très élevé. En fait, pour un botnet donné, plus le temps passe, plus il est raisonnable de s'attendre ce que les addresses IP qui le composent changent (certaines machines sont désinfectées, de nouvelles sont compromises : les adresses IP changent au cours du temps pour un même botnet. Donc, si le même botnet attaque nos plateformes à deux reprises à des périodes de temps distinctes, un moyen simple de les relier ensemble est en remarquant qu'elles ont une grande quantité d'adresses IP en commun. Plus formellement, nous mesurons la distance de deux macro attaques e_1 et e_2 comme la suitvante :

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

Nous dirons que e_1 and e_2 sont causés par les mêmes cause(s) (ou liés) si et seulement si $sim(e_1, e_2) > \delta$. Cela n'a de sens que pour une *valeur raisonnable* de δ.

Résultats

Nous avons identifié 30 (resp. 28) armées de zombies de AE-set-I (resp. AE-Set-II). Ils contiennent un total de 181 (resp. 234) d'macro attaques. La Figure 4 représente la distribution des tailles (en nombre d'attaques) des armées de zombies. L'histogramme du haut (resp. bas) représente la répartition des armées obtenues à partir AE-Set-I (resp. AE-Set-II). Nous pouvons voir que la plus grande armée a 50 attaques dans les deux cas, alors que 20 (resp. 18) armées n'ont été observés qu'à deux fois.

Analyse des armées de zombies

Durée de vie des armées de zombies

La Figure 5 représente la distribution cumulative des durées de vie des armées de zombies obtenus à partir de *AE-set-I* et *AE-set-II*. Comme on le voit, environ 20 % des armées de zombies ont une durée de vie de plus de 200 jours. Dans un cas extrême, deux armées semblent avoir survécu pendant 700 jours ! Un tel résultat semble indiquer que soit i) les machines compromises restent très long temps ou que ii) les armées sont capables de rester actives pendant de longues périodes de temps en compromettant en permanence de nouvelles hôtes.

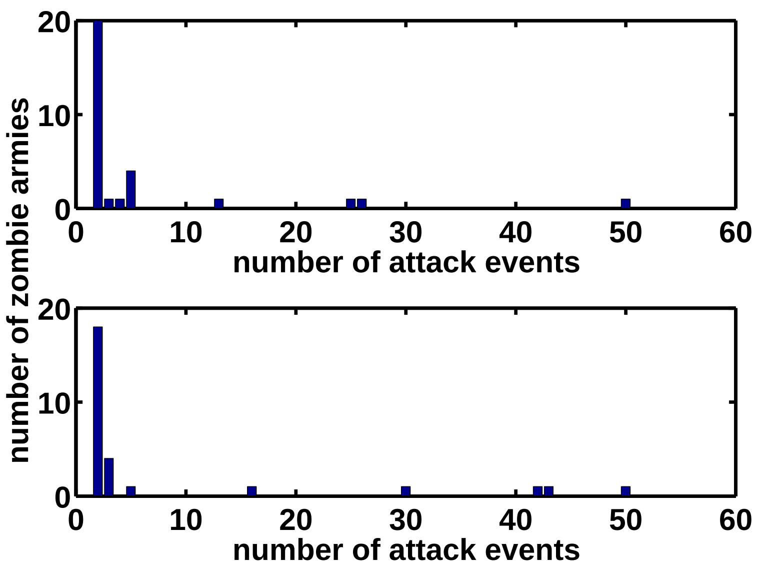


FIGURE 4 – Répartition des tailles d'armées de zombies

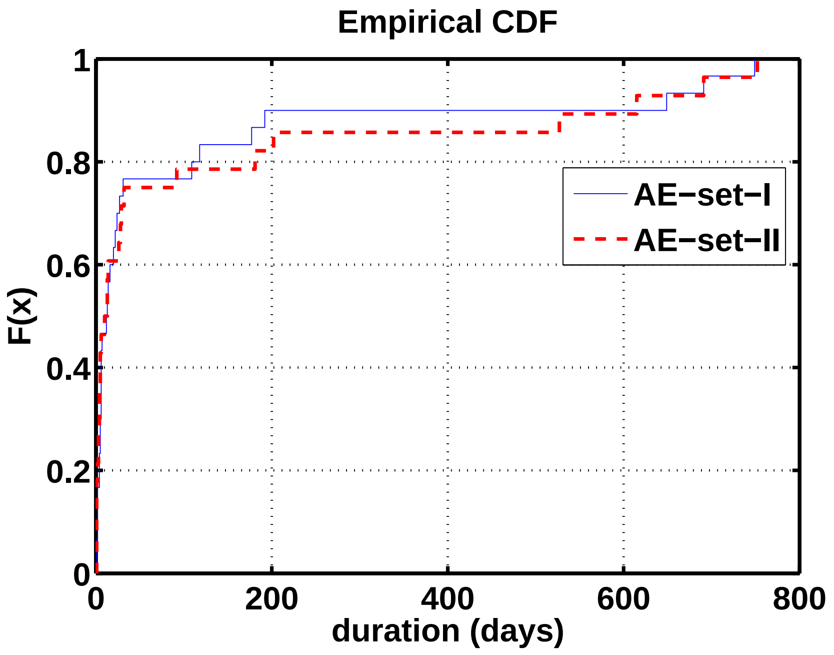
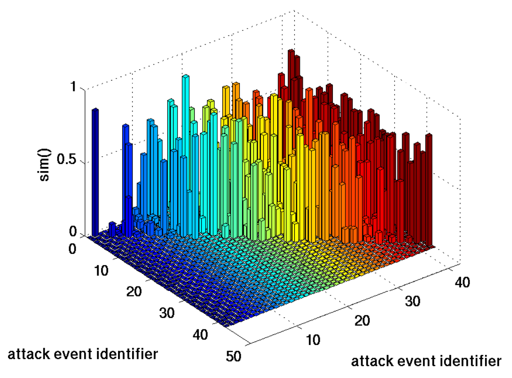


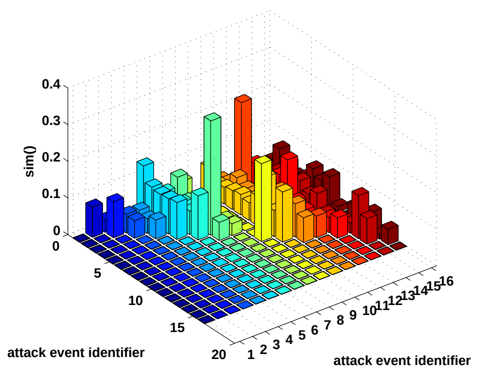
FIGURE 5 – CDF duration

Durées de vie des machines compromises dans les armées de zombies

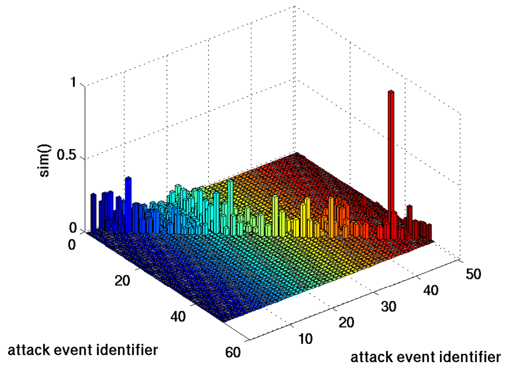
En fait, nous pouvons classer les armées en deux classes, comme mentionné dans la section précédente. Par exemple, la Figure 6 a représente la matrice de similarité de l'armée de zombie 27, AZ27. Pour construire cette matrice, nous trions d'abord les 42 macro attaques de cette armée en fonction du moment de leur apparition. Puis nous calculons leur similarité dans une matrice de la taille $42X42$ $\mathscr{M}$. La cellule (i,j) représente la valeur de $sim()$ entre la i^{ime} et la j^{ime} attaque. Puisque $\mathscr{M}$ est une matrice symétrique, pour des raison de lisibilité, nous n'en représentons que la moitié. Comme nous pouvons le constater, nous avons une très grande valeur de similarité , environ 60 %, entre presque toutes les attaques. Cela est également vrai entre les premières et les dernières attaques. Il est important de noter que l'intervalle de temps entre la première et la dernière attaque de cette armée était de 754 jours ! De même, la Figure 6 b représente la matrice de similarité de l'armée de zombies 24, AZ24. Cette armée de zombies a également une durée de vie très longue (625 jours), et tous les macro attaques partagent un nombre important d'adresses IP les uns avec les autres. Toutefois, dans AZ24, les valeurs de similarité sont beaucoup plus petits que ceux de AZ27.



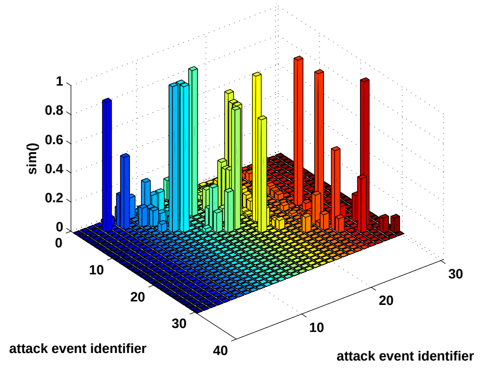
a) Zombie Army 27



b) Zombie Army 24



c) Zombie Army 28



d) Zombie Army 11

FIGURE 6 – Renewal rate of zombie armies

La Figure 6c représente un cas différent, la matrice de similarité de l'armée de zombies 28 (AZ28), composée de 50 macro attaques. Comme nous pouvons le voir, les valeurs importantes sont proches de la diagonale principale de (M) [1]. Cela signifie que deux macro attaques ne partagent un nombre significatif de machines sources que si elles sont proches l'une de l'autre dans le temps. Et l'existence de cette armée

1. La diagonale elle-même, bien sûr, n'est pas calculée depuis le $sim(i, i) = 1 \forall i$.

est de 536 jours ! Enfin, la Figure 6 d représente un exemple similaire (armée de zombies 11, AZ11) à ZA28, mais comme on le voit, les valeurs à proximité de la diagonale de la matrice de similarité sont beaucoup plus élevées. Il est clair, à partir de ces quatre cas, que la composition des armées évolue dans le temps de différentes manières. Ceci peut être vu grâce à l'identification des macro attaques et des armées de zombies mise au point dans cette thèse. Il reste cependant du travail à faire afin de comprendre les raisons derrière ces différentes stratégies.

Capacité d'attaque des armées de zombies

Par capacité d'attaque, nous faisons référence au nombre d'attaques différentes qu'une armée donnée est capable de lancer au cours de sa vie. L'histogramme du haut (resp. bas) de la Figure 7 représente la distribution de la quantité de clusters distincts par armées de zombies détecté à partir de *AE-set-I* (resp. *AE-Set-II*). Dans les deux cas, nous observons que la plupart des armées présentent des multiples traces d'attaque (correspondant à plusieurs clusters). Dans un cas extrême, une armée de zombies a plus de 120 cluster distincts. Ceci confirme ce qui avait été observé dans [69], "[...] Le botmaster semble demander à la plupart des bots dans un réseau de zombies de se concentrer sur une vulnérabilité, tout en choisissant un petit sous-ensemble de bots pour tester d'autres vulnérabilités". La Figure 8 représente l'évolution du nombre de clusters distincts dans le temps des AZ27 et AZ28. Dans les deux cas, le point (x, y) sur la courbe signifie que, jusqu'à la x^{ixime} attaque, on observe au total y clusters distincts. Comme nous pouvons le voir, dans les deux cas, les bots continuent à essayer de nouveaux clusters au cours de leur vie. Plus précisément, en observant plus en détail la courbe *armée zombie 27* dans la Figure 8 que AZ27 essaie un nouveau cluster presque dans chaque nouvelle attaque. La liste de séquences des ports correspondante qu'il a essayé est 5900/TCP, 135/TCP, 2967/TCP, 139/TCP, ICMP, 80/TCP, 445/TCP, 1433/TCP, 4899/TCP, 5901/TCP,18886T, 1026/UDP, 445T139T445T139T445T. Il s'agit d'un cas d'outil d'attaque avec multi-vecteur que l'on voit évoluer au fil du temps.

Exemple 1 : Armée de zombies 24, AZ24, est un exemple intéressant qui n'a attaqué qu'une seule plateforme. Toutefois, 16 macro attaques distincts sont liés à cette armée ! La Figure 9 présente ses deux premières activités correspondant aux deux macro attaques 56 et 57. La Figure 9 b représente quatre autres macro attaques. Dans chaque cas, l'armée tente un certain nombre de clusters distincts tels que 13882, 14635, 14647, 56608, 144028, 144044, 149357, 164877, 166477. Ces clusters essaient de nombreuses combinaisons de ports de Windows (135/TCP, 139/TCP, 445/TCP) et le serveur Web (80/TCP). L'intervalle de temps entre la première et la dernière activité est de 625 jours !

Exemple 2 : L'armée de zombies 27, AZ-27, est une autre grande armée que nous avons détectée. En fait, elle se compose de 42 macro attaques. À titre d'exemple pour montrer ses activités, la courbe en haut à gauche de la Figure 10 représente attaque 12 laissées par cette armée. La cible est la plateforme 26, et dans ce exemples, 85 machines d'attaque envoient des paquets ICMP sur trois pots de miel. La courbe à droite en haut montre macro attaque 307 qui envoie des paquets ICMP sur les

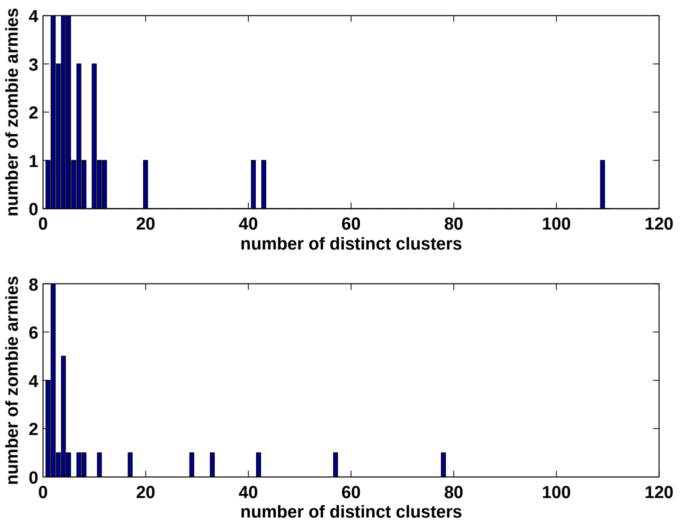


Figure 7 – zombie army Attack Capacity

Table 4 – Nombre des addresses IP en commun de quatre macro attaque 12, 307, 454, et 483

	12	307	454	483
12	85	48	60	36
307	48	93	63	37
454	60	63	370	47
483	36	37	47	108

deux plateformes 26 et 41. Dans ce cas en particulier, toutes les sources frappent seulement une seule adresse IP. On remarque que les deux macro attaques partagent 48 adresses IP. La courbe à gauche en bas de la Figure 10 montre l'macro attaques 454, cette attaque frappe 4 plateformes (26, 13, 50 et 57) et la dernière frappe 5 (26, 45, 53, 56, 57). Pour montrer à quel point ces macro attaques sont liés, la Table 4 montre le nombre d'adresses IP communes entre ces quatre macro attaques. Par exemple, lattaque 483 partage 37 adresse IP en commun avec l'attaque 307 et 454 et 483 partagent 47 adresses IP en commun... Finalement, l'intervalle entre le premier et le dernier événement d'attaque émis par cette armée de zombies est de 753 jours.

Conclusion

Cette thèse a commencé en disant qu'il est important de comprendre le modus operandi des processus d'attaque et que cette tâche est difficile. Les défis auxquels nous sommes confrontés pour résoudre ce problème sont dus manque de connaissances sur la manière et le moment où ces attaques sont lancées. Il est donc difficile de savoir si deux sources d'attaque sont liées (ou contrôlés par le même processus d'attaque) ou non. Nous avons besoin d'étudier les sources d'attaque dans leur

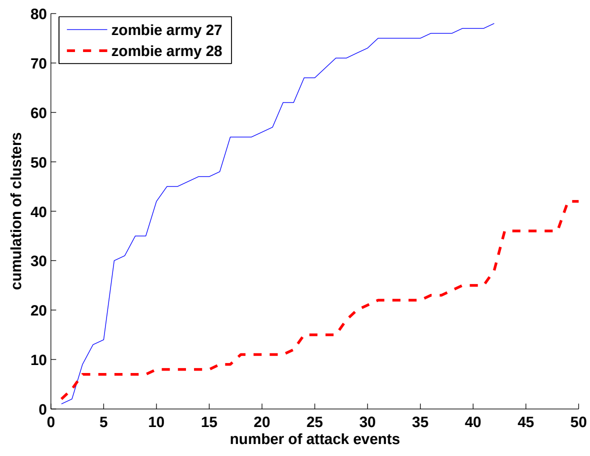


FIGURE 8 – Renewal rate of infected host of zombie armies

contexte afin de comprendre le processus d'attaque qui se cache derrière les exploits. Idéalement, nous aimerons pouvoir classer les machines d'attaque en groupes par leur cause originelle. Dans cette thèse, nous avons appelé de tels groupes de sources d'attaque des micro et macro attaques. Pour identifier et analyser les macro et macro attaques, nous avons proposé d'utiliser les techniques classique de traitement du signal et celles de fouille de données.

En ce qui concerne l'énoncé de la thèse qui a ouvert ce travail, nous avons montré qu'il est possible d'automatiser l'identification des micro et macro attaques. Pour ce faire, nous avons adopté les hypothèses clés suivantes : Les sources d'attaque ayant la même cause ont une distribution spéciale en termes de temps et de localisation géographique. Ceci nous amène à constater que les macro attaques existent sous forme de pics d'activités ou de groupes de traces d'attaques corrélées. Nous avons discuté et mis en place trois solutions alternatives afin de les identifier efficacement. Les solutions expriment le compromis entre le coût de calcul et la capacité à identifier les macro attaques. Toutes les solutions ont été soigneusement conçues pour fonctionner avec de nombreuses données. Comme la détection des macro attaques est basée sur la technique de corrélation des séries temporelles, nous avons étudié plusieurs mesures de similarité pour identifier la technique la plus adaptée à notre contexte d'application. Nous avons validé nos techniques sur un ensemble de données réelles collectées à partir d'un ensemble distribué de pots de miel. Et les résultats étaient conformes à nos attentes.

Comme une validation supplémentaire de notre approche dans la compréhension des traces des attaques, nous avons prouvé que les traces des attaques peuvent être classées en trois familles selon leur niveau d'activité. La première famille correspond aux outils d'attaque qui sont presque toujours utilisés dans le temps. La deuxième famille correspond aux outils d'attaque qui sont utilisés de temps à autre sur une période de quelques jours. La dernière famille correspond aux outils d'attaque qui sont rarement utilisés plus d'une fois et toujours pendant un ou deux jours seule-

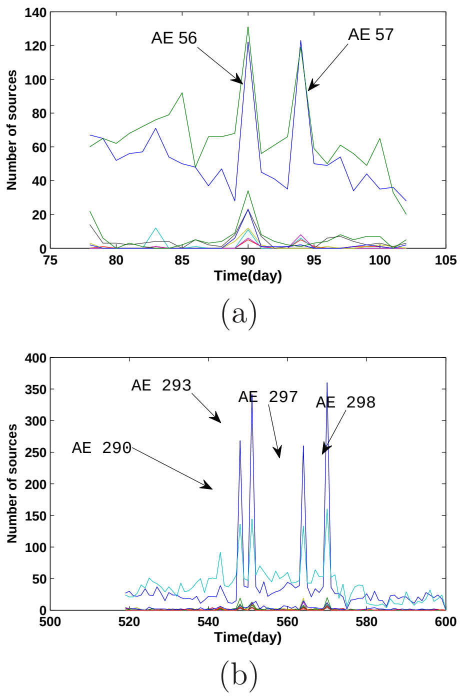


Figure 9 – 6 macro attaque de l'AZ24

ment. Outre cela, avec l'étude des macro attaques détectés par nos techniques, nous avons montré que nous pouvons identifier la cause et les caractéristiques de plusieurs macro attaques. En fait, à l'égard de la cause, les macro attaques peuvent être classés en trois catégories suivantes : *ver*, *botnet* et *autres*. En outre, nous avons proposé une méthode permettant de regrouper les macro attaques liées dans des groupes différents. L'étude de ces groupes révèle plusieurs aspects intéressants de ces menaces. Par exemple, dans le cas des réseaux de zombies, nous avons observé que certains botnets restent actifs aussi longtemps que 700 jours. Il est intéressant de voir que les botnets adoptent plusieurs stratégies pour renouveler leurs machines de zombies. Par ailleurs, certains réseaux de zombies ont réussi à survivre pendant une très longue période de temps même avec un taux de renouvellement de leurs machines de zombies extrêmement élevé. Nous avons également remarqué que les botnets changent fréquemment de vecteurs d'attaque. Il est important de souligner que nos solutions sont faciles à déployer et ne reposent sur aucune hypothèse concernant le protocole de communication utilisé par les botnets. Notre espoir est de donner un aperçu de la situation globale d'aujourd'hui (et d'hier) des activités des réseaux de zombies. Ce type de connaissance nous permettra d'avoir un regard plus précis sur les menaces actuelles auxquelles nos ordinateurs sont confrontés et donc de construire des contre-mesures plus efficaces.

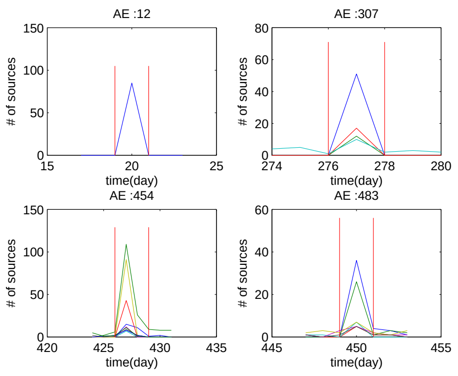


Figure 10 – 6 macro attaque de l'armée de zombies 27

Chapter 1

Introduction

1.1 Motivation

Having an accurate perception of the threat landscape on the Internet, and the ability to determine the attackers' identity and location is an area of research within the network security community. These topics are usually referred to as situational understanding and attack attribution [24]. The challenges we face when addressing these problems are that attackers try to obscure their identity. Hence, it is difficult to identify the primary controlling hosts. This process is especially true under the light of the endless innovation of attack tools. As an example, during the last six months of 2008, Symantec has discovered 1,656,227 new malicious codes [120]. This process is, foremost, motivated by the commercial gain from cyber crime activities [128]. Several business models have been discussed. In fact, there are many ways to get benefits from the compromised machines, for instance by stealing credential information, performing click fraud [52, 51], selling zombies machines [105, 39], using the compromised machines to send spam [6, 101, 126, 114, 53, 13], or doing DDoS attacks [54, 107, 72, 84]. As a result, the Internet has become an attractive place for many groups of attackers with different attack strategies, skills, and targets [40].

Among others, the threats found on the Internet have the following two characteristics :

– **The malicious activities have so called locality characteristics**. It has been shown that attacks observed on the Internet are not uniform neither in type nor in intensity [10, 22, 18, 28, 57, 81, 92, 91]. For instance, in [18] Pouget et al have provided a comparative survey of attacks observed against two identical sensors, one in France and the other one in Taiwan. The study has shown differences between the two attack traffics with respect to several aspects such as top attacked services, top domains of attacking machines, network-specific attack patterns. The locality property of attacks is also confirmed by other works [83, 22, 57]. Real worms Nimda [37], CodeRedII [75], Blaster [11, 15] are famous for their specific propagation mechanism. Also, by taking advantage of the information in BGP routing table, routing worms can not only propagate faster but are also able to choose the specific targets such as country, company, ISP, AS [139].

– **The compromised machines, organized into botnets, are more and more controllable**. A botnet is a network of compromised machines, called bots, that are under the control of their botmaster. While bots may share several characteristics with other classes of malwares, their distinguished characteristic is to use a command and control channel to receive commands from their master. Thanks to this property, attackers can exploit the compromised machines in different ways for different purposes. Botnets have been seen moving from a centralized infrastructure [12, 40, 51, 136] to a distributed one [44, 50, 116, 115, 113, 127], from easy to detect protocols [12, 40, 51, 136] to stealthier ones [20, 30, 51, 114]. According to [100], "[...] 27% of all malicious connection attempts observed from our distributed darknet can be directly attributed to botnet related spreading activity".

These trends highlight the fact that the dangers of an attack do not reside only in its exploit, i.e. in the actions taken to compromise a machine, but also in the way it is carried out (or the modus operandi employed by the tool). For instance, the locality property makes the attack tools stealthier and the controllability gives the attacker the ability to have various options on how to exploit the infected machines. While the modus operandi are important, they can not always be derived neither by observing the attack individually nor by analyzing its binary. This may be due to several reasons. The modus operandi of a tool is not always visible within a single attack. For instance, by studying individually the attack between one source and one destination, it is impossible to know whether it is a sweep scanning source. Also, information about the modus operandi does not always reside in the tools. For instance, in the case of a botnet, the targets of the attacks are received from the botmaster, they are not determined solely by the bots. Hence, we need to look at the attacking sources in their context to really understand the attack processes behind them, the strategies, and the motivation of the botmasters.

In the next section (Section 1.2), we introduce the research problem that we attempt to tackle in this thesis. And we present our contributions in Section 1.3. Section 1.4 describes the structure of the thesis.

1.2 Problem Definition

1.2.1 Thesis Statement

To assess the threat level of the Internet, researchers have taken several initiatives such as collecting malware samples [65, 8], analyzing malware dynamics [63, 26, 108], URLs assessment [99]. Collecting attack traces becomes an agreed needed task within the network security community [32, 77, 9, 27, 14, 36, 81, 96, 76]. Thanks to these collected data, analysts apply mathematical models to deduce characteristics of the current threats [2, 3]. For this approach to work, we need, among other things, a representative, clean dataset. In fact, a non-representative dataset may bias results. In the context of this thesis, we want to study the Internet security problem through the use of attack traces. As a groundwork to have sound results, we need a clean dataset as an input for our approach. In our case, to make it possible to assess the nature of the threats existing in the wild, we must collect the attack traces from different places in the world. Also, the attack traces generated by a given attack tool may depend on the environment it interacts with. Therefore, to be able to compare traces, they must be obtained thanks to the same sensor. As an attempt to understand the cyber attacks, since 2003, EURECOM has deployed a distributed honeypot sensor framework to observe and collect attack traces from different places on the Internet. As defined in [109], *a **honeypot** is an information system resource whose value lies in unauthorized or illicit use of that resource.* We will come back to this term in some more detail in the next Chapter. The long-term dataset collected from this infrastructure offers us the possibility to observe the evolution of the threats over a long period of time.

From the attacked network standpoint, we do not have any information about how the attacks are actually launched. It makes it difficult to explain the attacks we observe and to study the attack sources in their action context. In fact, they can be the mix of the interaction of several activities operated by different attackers. It is usually difficult to say whether two attacks are related or not. As a consequence, it is difficult to explain the attack phenomena where several attacking sources are involved. In the current state of the art in the field of attack trace analysis, most of the effort has been spent assessing the attack types [88, 81] and less on understanding how attacks happen [69]. Taking advantage of our distributed honeypot infrastructure, our intention is to group the attacking sources that we hypothesize of having the same cause and then study the characteristics of these groups to deduce the characteristics of the underlying threats. It is important to highlight that the cause of the attacks is not merely the type of the attack, but includes its particular use by a specific attacker. To detect such groups of attacking machines, we make the following key assumption. Attacking sources sharing the same cause will have a special distribution both in terms of time and space. Examples of such groups of attacking sources would be attacking machines in a botnet that are observed to attack a subset of our sensors during a period of time. In this case, by our expectation, the attacking machines should be grouped together. The machines, during that period of time, are members of what we call a micro attack event. More formally :

Definition 1 ***A micro attack event*** *μ is defined by a tuple $(\mathscr{T}, \mathscr{F})$ where $\mathscr{T}$ represents a limited period of time, typically a few days, and $\mathscr{F}$ represents the fingerprint of an attack as seen on our sensors. μ is a set of IP addresses observed over $\mathscr{T}$ and that have been seen leaving fingerprint $\mathscr{F}$ from a given observation viewpoint (e.g. one of our sensors)*

FIRST RESEARCH STATEMENT : *in this thesis, we want to build an automated mechanism that can recognize and characterize the micro attack events existing in malicious attack traffic collected from a distributed and similar honeypot sensors.*

Our assumption is that certain micro attack events may be linked together to make a larger attack phenomenon which is called a macro attack event.

Definition 2 ***A macro attack event*** *is a set of micro attack events observed over the same period of time and during which the corresponding time series are strongly similar.*

SECOND RESEARCH STATEMENT : *in this thesis, we want to build an automated mechanism that can recognize and characterize the macro attack events existing in malicious attack traffic collected from a distributed and similar honeypot sensors.*

1.2.2 Hypothesis

The attack traffic we observe on our honeypot sensors is made of raw packets. Statistical analysis of their features such as protocols, port numbers, number of packets can provide some level of indication of a new attack. For instance, one can detect things such as "there is an important increase of attacks on port X", "there is a peak of activities on protocol UDP on day Y". This kind of knowledge can help as an indicator for new attacks. However, further analysis is required to really understand what happens. Since the attack traces we observe on our network are due to several attack tools launched by different attackers, we must reduce the interference between several attack tools in order to analyze the threats. This comes down to being able to classify IP sources according to the attack they have been seen launching. Ideally, traces left by sources found in one class should be due to a single attack tool. This belongs to a larger research problem, i.e., traffic classification which is largely addressed in the literature [56]. Some works are focusing on the identification of attack tools by observing the traces they generate [66, 81, 89]. In our analysis, we do not address this research problem, but rather take advantage of previous results [89]. Therefore, we intentionally keep the discussion about this subject out of the scope of this work. This leads us to the following hypothesis.

HYPOTHESIS 1 *There exists a clustering algorithm that can distinguish the attack traffics generated by different attack tools.*

1.2.3 Approach and Challenges

To recognize the micro and macro attack events, we face the following issues :

– **Identification of micro attack events :** The first challenge we have is how to detect the micro attack events. Different attack processes may leave different fingerprints. For instance, one could argue that the traffic generated by botnets may be different from the one generated by worms. Our recognition ability may also be impacted by the way we cluster traces together. For instance, if a worm comes from a few specific places on the Internet, but attacks everywhere, it will be easy to identify the traces it leaves by grouping traces by the origin of the attacking sources. At the contrary, if a botnet consists of bots located all over the world that attack a few specific blocks of IP addresses, it is possible to highlight such specific behavior by grouping traces by destination instead of by source. The importance of the viewpoint in the analysis process will be discussed in Chapter 5.

- **Complexity :** As we will show later, to identify the micro and macro attack events, we need to compare the attack traces together. This is very CPU intensive task, especially for a large dataset. The factors that constitute this cost are :
  - The number of attack traces.
  - The length of the attack traces (or the time frame during which we collect the attack traffic).

  We offer a detailed analysis of this in Chapter 5.
- **Accuracy :** There exists many kinds of measures of similarity/distance such as SAX [70], Pearson Correlation, Minkowski. They were invented in different historical contexts and for different purposes. We need to identify the one that suits our purpose, which is to correlate the attack traces observed from the Internet. This is discussed in Chapter 4.

1.3 Contribution

Our contributions in this thesis include :

- We show that by analyzing the attack events we can understand more about the attack tools, the modus operandi of certain attack classes as well as several other characteristics. It is important to highlight the fact that the attack events generated as an output of our approach can be and, actually, have been used by other researchers in the context of their own work [122, 123].
- We demonstrate that it is feasible to automatize the detection of attack events for a large dataset. We develop three solutions for identification of attack events for different use cases. All three solutions have been experimentally validated.
- We show that attack tools can be classified into three families based on their activity level. The first family consists of attack tools that are almost constantly used in time. The second family consists of attack tools that are launched from time to time over a period of couple of days. The last family consists of attack tools that are rarely used more than once and always during one or two days only. This discovery is applied directly to reduce the computational cost when detecting attack events.
- We show the impact of the clustering of the attack traces on our ability to detect micro and macro attack events. Depending on whether we use the origin or the destination of the attacks, we may end up identifying different sets of attack events.

1.4 Structure of the Thesis

The remainder of the thesis is organised as follows :

In Chapter 2, we present several classes of existing infrastructures to collect attack traces on the Internet and provide a summary of several analysis that have been done in the field. This gives us the opportunity to clearly point out our contributions to the current state of the art.

Chapter 3 describes the Leurré.com infrastructure, an attack traces collection framework based on honeypot technology. We use data collected thanks to this architecture to validate our techniques. Although the techniques proposed in this thesis can be used on traces obtained thanks to other infrastructure than the Leurré.com one, due to the historical reasons, we adopt several notations that have been introduced in the past in the context of this seminal project. For the sake of completeness, they are also given in Chapter 3.

As discussed earlier, we need to compare the attack traces together, this means that we need some kind of similarity measure. The purpose of Chapter 4 is to first identify the properties that a similarity measure must have to enable us to compare the malicious attack traces. Then, we provide an in-depth discussion on the mathematical properties of several similarity measures. Based on this, we finally try to figure out the one that best suits our needs.

Chapter 5 builds upon the previous ones to offer some of the main contributions of this thesis. In that Chapter, we describe formally the problems we are facing and propose solutions. We also show experimentally the impact of the observation viewpoints. We validate our different techniques with a dataset collected from the Leurré.com infrastructure.

In Chapter 6, we justify the validity and the soundness of our method. We show, by analyzing the macro attack events detected in the previous chapter, that they enable us to see several characteristics of current botnets, or worms such as their lifetime and their attack vectors.

Finally, Chapter 7 concludes the thesis and provides the perspectives opened by this work.

Chapter 2

BACKGROUND AND RELATED WORK

This Chapter presents the state of the art in the field of attack traces analysis. We describe the most relevant works and try to establish the background on which we develop our thesis. To this end, Section 2.1 provides an overview of existing techniques and architectures to collect attack traces on the Internet. In Section 2.2, we summarize several kinds of analysis that have been done in the field of analyzing the malicious attack traces. Section 2.3 describes the various defense mechanisms that leverage the intelligence from attack traces collected. And finally, Section 2.4 discusses what is currently missing in the current state of the art to solve our problems. This discussion motivates the work done in the context of this thesis.

2.1 Attack Traces Collection

To build effective defenses, it is good to know one's enemies : the kind of tools they use, the tactics they employ, etc. With the similar idea in mind, network security analysts have started to collect attack traces left by the attackers to better understand them, and, thus, to be able to better protect their networks. Existing data collection infrastructures can be classified into the following two categories : production system and non-production system. We describe them hereafter.

2.1.1 Production System

The idea here is to gather security logs from several production systems. This approach is often referred to as log aggregation. The ability of such architecture to observe malicious activities is a fact of the sizes, distribution, and number of participating systems. Depending on the concrete system, security logs could be firewall logs, IDS security incidents, anti-virus logs. For instance, DShield [32], operated by SANS Institute, is one such project. By means of a web interface, voluntary log contributors can submit their firewall logs to the centralized database of DShield. DShield provides several high level statistics from the aggregated logs. Although sharing the same spirit, the approach taken by MyNetWatchman [77] is more automatized. In fact, it first asks participants to install its agent which is in charge of gathering the firewall incidents and sending them back to the MyNetWatchman's center server. To measure the dynamic of the attacking machines (IP addresses), log events are grouped by IP address. The system raises an alert to the owner of the specific IP address when the amount of security incidents from this address exceeds a certain threshold. All the feedback from the owners are documented in detail. DeepSight [119], run by Symantec, is an Internet scale framework with a similar data model. It asks the participants to install the so-called DeepSight Extractor, to gather the security logs. In exchange, DeepSight provides the reports about the attacks observed on the participants' computers. DeepSight collects data from a very large range of sensors : desktop antivirus, IDS, firewall... These data are used by DeepSight to predict the potential future threats that their client networks may face.

The advantage of this approach, as a means of data collection, is its ability to leverage the existing systems, with not much extra effort. However, the model has also some drawbacks. In fact, as indicated in [29], in case of DShield, there is a lack of the contextual information in the logs. As a result, the analysis based on these data may be biased or imprecise. Another downside of the approach concerns its inability to discover new kinds of attacks. In fact, in case of security incidents generated by signature based defense systems such as IDS, a classical anti-virus, the logs we have are all about known attacks (more precisely, attacks recognized by IDS, anti-virus). In case of firewall logs, we do not have much information about the attacks but rather about the violation of security policies.

Another approach, but related to this direction, is to extract flow information from routers [21]. As defined in [21] : "[...]A flow is identified as a unidirectional stream of packets between a given source and destination-both defined by a network-layer IP address and transport-layer source and destination port numbers[...]." Authors of [41] proposed a visualisation based method, what they called "contact surface", to visualize the amount of incoming connections to detect the outbreaks of Welchia.B worm. In a separated work, authors of [33] use flow information to detect the outbreaks of Blaster and Sobig.F.

2.1.2 Non-Production System

Monitoring attack traces shows that there is an important volume of network traffic targeting non-production IP addresses (or unused address blocks). Such traffic can be either malicious (flooding, backscatter, scans for vulnerabilities, worms) or benign (misconfiguration). We cover in this Section two classes of techniques that can be used to capture such non-productive traffic.

2.1.2.1 Darknet

A darknet or a network telescope is a large range of routable but unused blocks of IP addresses. It passively monitors the incoming traffics to these specific IP address blocks and does not generate any outgoing traffic. From the attacker's perspective, the presence of a monitoring can not be identified. On the one hand, it requires almost no effort to maintain a darknet, on the other hand, it provides little information about the attacks. In fact, for TCP based attack, the attacker must first establish the three-way handshake, before any payload can be sent. With darknet we can only observe the first TCP/SYN packet from which we may derive, at most, the service thanks to its destination port number. However, due to the lack of interaction with attackers, it is usually hard, if not to say impossible, to figure out the vulnerabilities that the attacks aim at. The major projects in this line are CAIDA Internet Telescope [14], Team Cymru Darknet Project [27], Internet Motion Sensors [9].

2.1.2.2 Honeypot Based Attack Traces Collection Framework

As discussed earlier, there is a need to engage and to maintain conversation with the attackers in order to be able to understand what their goals and strategies are. Honeypots have been created for this purpose. In fact, as defined in [109] by Spitzner in 2002, *"A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource"*. According to this definition, a honeypot can be a machine with a real operating system that is plugged to the Internet and wait for being attacked. It is important to note that, this definition corresponds to the traditional honeypot. In this case, it acts as a server, exposing some vulnerable services and passively waiting to be attacked. This class of honeypot may not be able to detect client side attacks. In fact, to detect client side attacks, a system needs to actively interact with the server or process malicious data. Another type of honeypot is therefore needed : the client honeypot. Client honeypots crawl the network, interact with servers, and classify servers with respect to their malicious nature [133, 80]. In the context of our work, we limit ourselves only on studying traffics issues from the traditional honeypot. As a convention, from now on, we use honeypot to refer to the traditional honeypot. Based on the level of interaction, we can classify honeypot as low, medium and high interaction. As the illustrated examples, Honeyd [98], Nepenthes [8], HoneyTank [125], iSink [130], Billy Goat [103] are considered as low interaction honeypot. Whereas, ScriptGen [66] is a medium interaction honeypot. Roo Honeywall [97], BotHunter [46], Honeybow [137] are examples of high interaction honeypot. In a simple word, the higher level of interaction a honeypot possesses, the more functionality it has (or the more it acts as a real host). And thus, the more information we can get from and about the attackers. But there is a trade-off between the level of interaction and the security of a honeypot. In fact, high interaction honeypots give rich information but suffer also from a increased level of security risk. Adding on top of that, it is usually more costly to deploy high interaction honeypots. Based on these technologies, several data collection frameworks have been deployed [36, 81, 96, 76, 10].

2.2 Understanding Threat Landscape of Internet

The final purpose of collecting attack traces is to understand the threat landscape of the Internet. However, this task is not evident due to the extremely large amount of attack traffic that can easily overwhelm network security analysts. Several approaches have been proposed to analyze the malicious attack traces automatically. They are described hereafter.

2.2.1 Activity Observation

2.2.1.1 Observation of Denial of Service Attacks

This class of analysis aims at studying a specific aspect of the threat landscape of the Internet. In [74], Davide Moore et al have proposed a method to measure the prevalence of DDoS attacks on the Internet thanks to data collected by Caida's Internet Telescope. The authors assume that flooding attacks are the most popular DoS attacks. These attacks aim at consuming the resources of the victim, otherwise, used by benign client requests. The idea is that the attacker sends an important amount of packets to overwhelm the victim's CPU, memory or network resources. And this makes the victim unable to answer the requests coming from legitimate clients. To conceal the origin of the attacking machines, attackers usually forge, or spoof the source IP addresses of the packets. Therefore, from the victim stand point, the packets appear to come from the third parties. In the case that the third party addresses fall within the monitored IP address blocks, they can be used to infer the prevalence of DDoS attacks. Taking advantage of data collected from Network Telescope [14], in this analysis, the authors have provided diverse information on several facets of DDoS attacks such as victim classification, attack duration, attack rate, etc.

2.2.1.2 Worm Propagation and Botnet Activity

Darknets cover usually large portions of IP space. If a large scale worm outbreak occurs, there is a large chance that it hits the darknets. By monitoring the darknets, we can derive several characteristics of worms such as their method and speed of propagation. This has been demonstrated during the CodeRed and Slammer worm cases [75, 73]. Besides that, the data collected from darknets have also been used to build and/or validate worm models [110, 111].

There is a consensus in the security community to say that botnets are today's plague of the Internet. A lot of attention has been paid to detect and eradicate them. Several approaches have been proposed for this purpose. By identifying the so called *Command and Control (C&C)* channels, one can keep track of all IPs connecting to it. The task is more or less complicated, depending on the type of *C&C* (IRC [23, 40, 12, 51, 43, 100], HTTP [20, 30, 114], fast-flux based or not [49, 82, 102], P2P [50, 44, 115, 127], etc.) but, in any case, one needs to have some insight about the channels and the capability to observe all communications on them. Another approach consists in sniffing packets on a network and in recognizing patterns of *bot-like* traffic. This is, for instance, the approach pursued by [46, 45, 47] and [117, 112]. The approach proposed in [69] aims at detecting botnets by observing their probing events left on monitored networks (darknet, and honeypot). In fact, the authors assume that "most probing events reflect activity from the coordinated botnets that dominate today's Internet attack landscape". To detect the probing events, the authors first classify attack traces into sessions as proposed in [55]. On the session basis, the amount of sources arriving within a time interval are computed. If this number is above a certain level computed from the previous steps, this is considered as an unusual event. Unusual events are then classified into three families : worm, botnet probing, and misconfiguration. The paper then provides a set of scan patterns checking to determine the scan strategies used by botnets, and then uses the botnets that have the uniform scanning patterns to extrapolate the attacks of botnets at the Internet scale. While the take away message is different, this work shared a large portion with the one in [129]. We present this work in more detail in Section 2.3.

2.2.2 Identifying the Type of Attacks : The "What" Question

In this Section, we present the class of analysis or framework that tries to answer the question : "what attacks are we facing ?". This includes the qualitative and quantitative analysis of types of the attacks from the Internet. It focuses more on the understanding of the attacks at a low level and less on the attack strategies.

2.2.2.1 Activity Report

Sometimes, it is interesting to have the very general, and easy to obtain, statistics about the attack traces. For instance, one of the very basic questions is to know what services are attacked the most. This information may offer clues on new attack phenomena, for example, new worm outbreaks. Several services provide such information, as an example, DShield web interface [32] provides several high-level statistics such as top attacked port, top attacking country, top attacking sources. Team Cymru [27] provides the distribution of attacking sources on the IP space. It is important to notice that, such statistics can only be used to provide hints about the new phenomenon, further analysis is needed to determine the true nature of the attacks. For instance, in the case of Blaster, after sending the attack on port 135/TCP to force the vulnerable machine to open a backdoor on port 4444, the attacking machine sends packets to port 4444. As a consequence, during the Blaster outbreak one could observe two distinct phenomena, namely an increased traffic against port 135 and also against port 4444. However, these two events had the same root cause.

2.2.2.2 Attack Type Analysis

In this Section, we present the more in-depth analysis techniques that aim at recognizing and quantitatively measuring various types of attacks taking place on an Internet wide scale. One of the representative work in this line is the one done by

Pang et al in [81]. With the motivation of understanding the nature of background radiation of the Internet, the authors have proposed a framework that measures the Internet activities as seen from a very large portion of IP addresses. They use these traces to recognize several types of attacks. The method consists of two steps. In the first step, they build an effective filter that significantly reduces the amount of traffic while keeping the diversity. Then they build a large responder framework that can handle a huge amount of IP addresses. To achieve this, the authors rely on the iSink technology [130]. iSink is a stateless responder which is in contrast to the approach taken by Honeyd. These responders are responsible for maintaining the conversation with the attackers up to a point where attacks are distinguishable. Obviously, doing this for all the incoming connections is very expensive in terms of computational cost. To decide which TCP session should be extended, a data-driven approach is adopted : the most common form of traffic is selected for building the responder, and the process repeats until the unknown traffic is sufficiently small.

With a similar idea, i.e., to recognize the type of the attacks that honeypots are facing, in [89] Pouget et al have chosen a clustering based approach. In fact, the authors have proposed a method to cluster the attacking sources based on the fingerprint left on a set of honeypots. To do this, authors start first by withdrawing the network influences such as packet reordering, and packet losses based on IPID. Then, by combining the network parameters and the configuration of honeypots. Seven attributes are chosen to discriminate the different attack types. This includes the number of packets sent by a source, the number of honeypots hit, the sequence of destination ports, the duration of the attack, the average inter arrival time of packets, the average number of packets sent to each honeypot, the payload (if any). Then a non supervised learning approach is applied to classify distinct activities into different clusters. It is obvious that the challenge of this approach is to choose a good set of features for the clustering. We provide a more detailed description of the algorithm in Chapter 3. The validity and usefulness of this clustering approach has been discussed and applied in previous publications [93, 138, 18].

ScriptGen [66] is a honeypot that is able to identify and handle new attacks automatically. It can, therefore, be used to observe and classify the attacks. In fact, according to [67], "[...] The ScriptGen technology was created with the purpose of generating honeypots with a high level of interaction having a limited resource consumption. This is possible by learning the behavior of a given network protocol when facing deterministic attack tools. The learnt behavior is represented under the form of a Finite State Machine representing the protocol language. The generated FSM can then be used to respond to clients, emulating the behavior of the real service implementation at a very low cost[...]." Furthermore, ScriptGen has also the capacity of handling the yet unseen attacks. Indeed, if a client request falls outside the current FSM knowledge (there is no matching transition), it forwards all the clients requests to a real host and then acts as a proxy between the attacker and the real host. The way a real host deals with such attack is then observed and learned. This knowledge is then used automatically to refine the protocol knowledge and to push new nodes into the FSM present in each sensor [65]. Although there are differences in the details, work developed in parallel and presented subsequently in

[25] shares main conceptual ideas with this work.

2.2.3 Inferring Internet Properties : The “How” Question

Another research direction is to understand and/or induce the general properties of threats found on the Internet. At some level, these properties reflect the attack strategies used by the attack tools, the analysis aims at answering how the attack happens. For instance, work in [5] studies the evolution of the scans on the Internet over a very long period of time. The authors have shown that most of the attacks on the Internet have moved from direct attacks to scanning. Also in an attempt to understand the distribution of attacks in the IP space, several research initiatives compared the attack traces collected from different places on the Internet and have shown that the attacks have the locality property mentioned before. Namely, it has been reported that the attacks are different both in volume and type from place to place [22, 18, 81, 92]. Also, when analyzing the behavior of the infected machines on their target, authors of [57] have stated that “...20% of all offending sources mount correlated attacks and they account for more than 40% of all the IDS alerts...”. Furthermore, attacking machines involved in correlated attacks can be grouped into clusters by looking at their targets. In other words, there are several sets of attacking machines, and each group has its own target profile. Futhermore, as claimed in [57], the list of targets is often very limited. In fact, when using logs collected from 1,700 IDSs, the authors of [57] have concluded that : “[..] 1700 IDSs can be divided into small groups with 4-6 members that do not change with time ; IDSs in the same group experience a large number of correlated attacks, while IDSs in different groups see almost no correlated attacks.” The correlated attacks are also confirmed in [81]. Another class of attacks are the coordinated attacks launched simultaneously by a set of sources. Examples of this are botnets’ activities [69, 129, 81].

In [121], Thonnard et al have proposed a framework to analyze the attack phenomena on a multi-viewpoints approach. The purpose of this ambitious approach is to figure out properties that several attack phenomena could have in common.

Finally, there is a class of works that is at the same time similar and different from both categories mentioned before. In fact, they do not attempt to recognize all the attacks but just some attack patterns and, then, analyze their properties. For instance, authors of [138] show that the inter-arrival time of packets can be used to detect certain class of activities. Pouget et al have shown in [93] how to use time series correlation to detect the multi-headed attack tools. When studying the persistence of worms, the authors of [131] concluded :”[...]We also find that worms like CodeRed, Nimda and SQL Snake persist long after their original release[...].” In the same paper, when studying the behavior of attacking sources, the authors observed :”[...]a very small collection of sources are responsible for a significant fraction of intrusion attempts in any given month and their on/off patterns exhibit cliques of correlated behavior[...].”

2.3 Threat Defense

In [129] Yegnesvaran et al have proposed a framework to integrate the intelligence learnt from honeypot traces to help network security analysts. The idea is to provide some information about the current threats that could help network security analysts when making decisions. This information could be hints on the cause of the attacks (e.g., a new worm, a botnet, or a misconfiguration), or whether the attacker specifically targeted the victim network, and if this attacker matches the one seen in the past. The process is done as follows. The network activities are classified by using Bro, and then based on this, by means of statistical measures, the unusually large-scale events are detected. Based on the shape of attack traces, the traffic are then classified into three families : worm, botnet, or misconfiguration. The main challenges of the work is how to write the Bro policies to define the attack profiles. In the same line but more generic, Allman and colleagues [4] propose to leverage "[...] the deep understanding of network detectives and the broad understanding of a large number of network witnesses to form a richer understanding of large-scale coordinated attackers". With similar ideas, work in [68] tries to build an architecture that can index the monitored data from a distributed system : "[...] Such systems consist of two logically distinct components : a distributed network monitoring system, and a distributed querying system". The detectives in this case are distributed querying system and witness are distributed monitoring system. Work in [17] can be classified along the same line as the two previous ones.

Zhang et al [135] argued that the two widely adopted blacklisting strategies (*global worst offender list* (GWOL) and *local worst offender list* (LWOL)) are not optimal under the light of current threats on the Internet. Actually, the GWOL uses the large scale repository to build the blacklist and it will be used by network administrators to secure their networks. The authors argued that such list can not capture the more strategical attackers, focusing on a few known vulnerable network [19]. As opposed to GWOL, LWOL uses the enterprise access log history to form the address blacklist for itself. While LWOL can capture most of the frequent attackers, it is reactive. In fact, it can only capture known IP addresses, most of the time, after the fact. Results in [57] show that, in the current state of Internet, there are clusters of networks that seem to face a common set of attacking sources. To address the issues of both blacklist strategies, Zhang et al [135] suggested a new strategy for blacklisting called *highly predictive blacklisting*. The blacklist is formed based on a large number of access logs from different networks. The administrator form the blacklist based on both the amount of attacks and networks being attacked. Indeed, if an attacking source attacks network(s) in one cluster, there is a large chance that it will attack the other networks in that cluster too. Therefore, the corresponding IP address should be added into blacklists that applied for all networks in that specific cluster. Since this attacking source does not seem to attack the networks different than the ones in the mentioned cluster, it does not bring much sense to add it to their blacklists.

2.4 Discussion

This Chapter has offered an overview of several types of analysis that can be done with data collected from monitored network, with a particular focus on honeypot traces. Echoing to the motivation we have pointed out in Chapter 1, our work has similarities but is also different from the ones in [129, 69]. In fact, in the purpose, we all try to identify the attack events, groups of attacking machines acting under the same cause. To detect the attack events, authors of [129, 69] examine the attacks at different places separately. In our case, we go one step further, i.e., we use the attack information from several places to identify the attack events. To detect attack events occurring on several locations, we need a totally different set of techniques from the ones used in [129, 69]. They are described and applied in the rest of this thesis.

Chapter 3

SOURCE OF INFORMATION

3.1 Introduction

EURECOM has started collecting attack traces on the Internet since 2003 by means of honeypot responders. The first platform consisted of three high interaction honeypots built on top of the VMware technology (the interested reader in the platform configuration is invited to read [90] for more information). As shown in [90, 28], these first experiments allowed us to detect some locality in Internet attacks : activities seen in some networks were not observed in others. To validate this assumption, we decided to deploy multiple honeypots in diverse locations. With diversity, we refer both to the geographical location and to the sensor environment (education, government, private sectors, etc). However, the VMware-based solution proved not to be scalable. First, this solution had a high cost in terms of security maintenance. Second, it required significant hardware resources. In fact, to avoid legal issues we would have needed to ensure that these systems could not be compromised and could not be exploited by attackers as stepping stones to attack other hosts. For those reasons, we have chosen a low-interaction honeypot solution, honeyd [98]. This solution allowed us to deploy low-cost platforms, easy to maintain and with low security risk, hosted by partners on a voluntary basis. The low-cost of the solution allowed us to build a distributed honeynet consisting of more than 50 sensors distributed all over the world, collecting data on network attacks and representing this information under the form of a relational database accessible to all the partners. Information about the identity of the partners and the observed attackers is protected by a Non-Disclosure Agreement signed by each entity participating to the project.

The primary purpose of this chapter is to present the data collection framework used to validate the approach proposed in this thesis. We also want to explain the practical motivation that makes our infrastructure as it is now. This includes how to manage a large distributed system and the key design rationales of our database schema that enables us to fulfill our operational requirements.

3.2 Leurré.com : Data Collection Architecture

We describe here some important technical aspects, including the platform architecture, the logs collection mechanism, the DB uploading mechanism, and the data enrichment mechanism.

3.2.1 Platform Architecture

As mentioned before, the main objective is to compare unsolicited network traffic in diverse locations. To make sound comparisons, the platform architecture must be the same everywhere. We tried to make our Honeyd-based solution as similar as possible to the initial VMWare setup [90]. We configured Honeyd to simulate 3 virtual hosts running on three different (consecutive) IP addresses. We configured Honeyd's personality engine to emulate the presence of two different configurations, namely two identical virtual machines emulating Windows 2000 SP3, and one machine emulating a Linux Kernel 2.4.20. To the first two configurations (resp. the last) correspond a number of open ports : FTP, Telnet, Web server, Netbios name service, Netbios session service, and Service Message Block (resp. FTP server, SSH server, Web server on ports (80), Proxy (port 8080, 8081), remote shell (port 514), LPD Printer service (port 515) and portmapper. We require from each partner hosting the platform a fourth IP address to access the physical host running Honeyd and to perform maintenance tasks. We run tcpdump [95] to capture the complete network traces on each platform. As a security measure, a reverse firewall is set up to protect our system. That is, we accept only incoming connections and drop all the connections that could eventually be initiated from our system (in theory, this should never happen). The access to the host machine is very limited : SSH connections are only allowed in a two-hours daily time frame and only if it is initiated by our maintenance servers.

3.2.2 Data Collection Mechanism

An automatized mechanism allows us, on a daily basis, to connect to the platforms through an encrypted connection to collect the tcpdump traces. The script, thanks to the diary on the central server, downloads not only the last day's log file but also, possible, older ones that could not have been collected in the previous days due to, for example, a connectivity problem. All the log files are stored in a central server. Since platforms could be down or inaccessible for many reasons, another automated script checks the data collection status, identifies and sends the list of need-to-be-checked platforms to the administrator for him to take the relevant actions.

3.2.3 Data Uploading Mechanism

Just after the data retrieval, the log files are then uploaded to a large database (built on top of Oracle) by a set of Perl programs. These programs take tcpdump files as input and parse them in order to create different abstraction levels of the attacks. The lowest one corresponds to the raw tcpdump traffic. The higher level is built on top of the lower ones and has richer semantics. We present some important concepts in the next section.

3.2.4 Information Enrichment

To enrich the information available about each source, we add to it three other dimensions :

1. **Geographical information** : To obtain geographical location such as organisation, ISP, country of a given IP address, we have initially used Netgeo [94], developed in the context of the CAIDA Project. It provided a very surprising result which flagged Netherlands and Australia as the two most attacking countries. As a sanity check, we have used Maxmind [71] and we have detected problems with the Netgeo classification. [91] provides a comparison of these two tools. As a result, we have kept using the most reliable one, namely Maxmind.
2. **OS fingerprint** : To figure out the OS of attacking hosts, we have used passive OS fingerprinting techniques. We take advantage of disco [106] and p0f [134]. Active fingerprinting techniques such as Nmap, Quezo, or Xprobe have not been considered because we wanted to minimize the risk of alerting the attacker of our investigations.
3. **Domain name :** We also do reverse DNS lookups to get the domain name of the attacking machines (when available).

3.3 Leurré.com : An Abstract Level of Database Schema

3.3.1 Design Rationale

The purpose of an Entity-Relationship model is to allow the description of the conceptual scheme based on a practical relational model. We can usually optimize this process with respect to various types of constraints (speed, memory consumption, table sizes, number of indices, etc.). Best practices in designing relational databases address problems such as data redundancy and update/insertion/deletion anomaly issues. In database terminology, the update/insertion/deletion anomaly is the update/insertion/deletion operation that brings the data into the inconsistency status. To address this issue, people usually apply a so-called normalization process [58] to bring the database into the “normal-forms”. The higher level the normal form is, the better it is in terms of avoiding redundancy, and, thus, of introducing anomalies during the operational lifetime. To do this, tables that do not satisfy the normal-form property are usually split into smaller ones. However, when searching for information, we usually need to join several tables together, and normal forms may be less efficient in these cases. In our case, we follow this design principal wherever possible, but it is not an obligation. The reasons for that are twofold. Firstly, we do not care that much about update and insertion anomaly issues as we do not usually modify the data once it is inserted into the database, as it represents a fact of the past. Secondly, we do care a lot about the efficiency of querying the database and we are not so concerned by space efficiency (be it on disk or in memory) as the total amount of data we deal with remains rather modest, compared to what existing database systems can handle. Therefore, we have consciously decided to integrate in our design some redundant information. In other words, certain tables contain information that could be retrieved by querying or joining other tables. However, having the results of such queries available at hand in ad-hoc tables proves to be extremely useful when using the database. As a consequence, we decide to keep them, acknowledging the fact that, without their presence, the database would be more 'optimal' according to the classical criteria.

3.3.2 Database Description

3.3.2.1 Main Components

The database schema can be divided into four main components as follows :

- **Platform Information :** As the name suggests, this component holds information about the platform such as its IP addresses, geographical location, the time zone.
- **Packet Information :** This component consists of several large tables related to the attack traces at packet level. In theory, we can reconstruct raw packets from these tables. As a matter of fact, they are considered as the original source of the information or a "backup" of other tables.
- **Contextual Information :** Various information about the attacking machines are added to enrich our views on the attackers. It includes the geographical locations, the domain names, and the operating systems.
- **Meta Data :** From the packet information, we build many concepts that represent the attacks at higher semantic levels.

3.3.2.2 Meta Data

As mentioned earlier, we build several concepts that represent the attacks at a richer semantic level. For the ease of understanding the next chapters, we present some important ones in this section.

Definition 3 ***Source*** *: A source corresponds to an IP address that has sent at least one packet to, at least, one platform. Note that, in our Source model, a given IP address can correspond to several distinct sources. That is, an IP remains associated to a given source as long as there is no more than 25 hours between 2 consecutive packets received from that IP by any platform. After such a delay, a new source will be assigned to the IP. By grouping packets by sources instead of by IPs, we minimize the risk of gathering packets sent by distinct physical machines that have been assigned the same IP dynamically after 25 hours.*

Definition 4 ***Ports sequence*** *: A ports sequence is a time ordered sequence of ports (without duplicates) a source has contacted on a given virtual machine.*

For example, if an attacker sends the following packets : ICMP, 135 TCP, 135 TCP, 139 TCP to a given virtual machine, the associated ports sequence will be represented by the string $I|135T|139T$.

This is an important feature that allows us to classify the attacks into different classes. In fact, as mentioned in [89], most attack tools are automatized, it is therefore quite likely that the same attack tools will leave the same port sequences on different platforms.

Definition 5 ***Tiny Session*** *: A Tiny Session groups the packets exchanged between one source and one virtual host.*

Definition 6 ***Large Session*** *: it is the sequence of packets that have been exchanged between one Source and a particular honeypot sensor. A Large Session is characterized by the duration of the attack, the number of packets sent by the Source, the number of virtual machines targeted by the source on that specific platform. A Large Session is thus composed of up to three Tiny Sessions.*

Definition 7 ***Cluster*** *: A Cluster is a set of similar large sessions observed on any possible platform.*

We apply the clustering algorithm defined in [89] on the traffic generated by the sources. The first step of this clustering algorithm consists in grouping large sessions into bags. This grouping aims at differentiating between various classes of activity taking into consideration a set of preliminary discriminators, namely the number of targeted virtual hosts and the unsorted list of port sequences hitting them. In order to further refine the bags, a set of continuous parameters is taken into consideration for each large session, namely : its duration, the total number of packets, the average inter arrival time of packets, and the average number of packets sent to each honeypot. These parameters can assume any value in the range $[0, \infty]$, but some ranges of their values may be used to define bag subclasses. This is done through a peak picking algorithm that identifies ranges of values considered discriminatory for the bag refinement. Large sessions belonging to a bag and sharing the same matching intervals are grouped together in a cluster. A very last refinement step is the payload validation. The algorithm considers the concatenation of all the payloads sent by the attacker within a large session ordered according to the arrival time. If it identifies within a cluster multiple groups of large sessions sharing similar payloads, it further refines the cluster according to these groups. In summary, a cluster is by design a set of large sessions that seem to be originating from a similar attack tool.

Figure 3.1 represents the relationship of some of the most important concepts in the Meta Data component. As we can see, a *Source* may have from one to n *Large Sessions*, n being the number of platforms. A *Large Session* may have from one to three *Tiny Sessions*, and one *Large Session* belongs to one and only one *Cluster*.

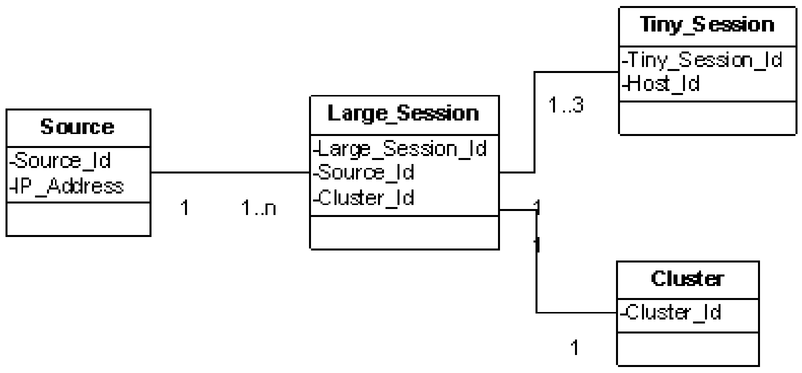


FIGURE 3.1 – Some important concepts in the Meta Data component

3.4 Generic picture

Our first platform has started running in February 2003, since then new partners have joined our collection framework and, therefore, the volume of data kept increasing month after month. Some global statistics are listed as an indication of the important volume of data we have now.

- Number of platforms : 50
- Number of observed distinct IP addresses : 3,800,791
- Number of distinct Sources : 5,863,674
- Number of received packets : 354,971,435
- Number of emitted packets : 284,580,993
- Number of received TCP packets : 331,513,017
- Number of received UDP packets : 14,935,446
- Number of received ICMP packets : 9,761,973
- Number of distinct Tiny_Session : 11,290,893
- Number of distinct Large_Session : 6,742,980
- Number of distinct port sequences : 218,107
- Number of distinct clusters : 156,234

3.5 Conclusion

We have presented in this Chapter several aspects of our data collection framework. It includes the platform configuration, the data collection mechanism, the data uploading and enrichment. We have also offered a general view of our main database components as well as their rational design. Some important concepts have also been described.

Chapter 4

TIME SERIES CORRELATION TECHNIQUE FOR MALICIOUS NETWORK TRAFFIC

4.1 Introduction

Searching for similarity is an important problem when mining time series. In fact, it is part of almost any classical data mining problem : clustering [59], indexing [1, 86, 38, 16, 132], classification [42]. In general, to know the distance between two time series X and Y, people apply a distance function $\mathscr{F}$ either directly on raw data X and Y, or on X' and Y' (X' and Y' are derived from X and Y, respectively, with $|X'| \ll |X|$ and $|Y'| \ll |Y|$). The different distance functions have been invented and used for different situations. In our case, we need to compare the malicious attack traces to discover the coordinated attack phenomena existing in those traces. Given this application context, we have certain constraints on the properties of the distance function. To identify the right distance function for our case, in this Chapter, we first study the mathematical properties of a set of representative similarity measures for time series data. Then, with respect to our application context, we enumerate a set of requirements that the expected distance function must satisfy. Based on this foundation, we choose the one that best suits our needs.

The remainder of this chapter is organised as follows. Section 4.2 briefly discusses the notations of distance and similarity function. We present some examples of real attack traces in Section 4.3. Section 4.4 presents different time series distance functions. We discuss the applicability of these distance functions in the context of correlation of malicious attack traces in Section 4.5. Finally, Section 4.6 concludes the Chapter.

4.2 Distance and Similarity Measure

So far, we have used the term distance and similarity function indifferently, but in reality, they are two distinct notions. This Section makes a clear distinction between the two.

In mathematics, a distance function (or a metric) on a given set M is a function $d : M \times M \to R_+$ that has the following characteristics :

1. *non-negativity* $d(x, y) \geq 0$ and *Identity of indiscernibles* d(x,y)=0 if and only if x=y
2. *symmetry* d(x,y)=d(y,x)
3. *triangle inequality* $d(x, z) \leq d(x, y) + d(y, z)$

The distance is used to measure how far the two points are from each other in the space M. The more important the distance is, the further the two points are, and vice versa. Given our application context, i.e., to compare the attack traces, we do not expect that the distance measure we use satisfies all the above conditions. For instance, as we argue in Section 4.5, some activities may be more important than the others in the attack traces, it is not necessary for two attack traces to be identical to have the distance of zero. In this case, we may not need the *identity of indiscernibles* property (d(x,y)=0 if and only if x=y). As a convention, from now on, our notion of distance is referred to this new sense.

Contrary to distance, the similarity measure assesses how close (similar) the two points are. We can convert the distance unit to similarity unit. This can be done as follows :

$$s(x, y) = MAX - d(x, y) \tag{4.1}$$

in which, x and y are two points in the space M. s(x,y) stands for the similarity between x and y. MAX is the theoretical maximum distance between two points in M, and d(x,y) is the distance between x and y.

4.3 Examples

We present three examples of real attack traces observed on our infrastructure and we use them in the rest of this Chapter to show the needs and constraints we have when looking for a good similarity measure.

The example in Figure 4.1a represents the evolution of attacks of cluster 156423 on two platforms, 7 and 29, over a period of 151 days. As we can see, the waves of attacks against these two platforms are highly synchronized. It is quite likely that the attacks on these two platforms have the same underlying cause. Therefore, we want the similarity measure to conclude that these attack traces are correlated. Figure 4.1b shows the attacks of cluster 14647 against the platforms 34 and 39 over a period of 31 days. As we can observe, the attacks on platform 39 consists of two strong peaks of activities on days 473 and 480 whereas the attack on platform 34 does not have the corresponding peaks of activities. For this reason, we expect that the similarity measures flag these attack traces as non correlated. Finally, Figure 4.1c represents the attack traces of cluster 15715 against the platforms 33 and 37 over a period of 41 days. Each attack trace has a major peak of activities but not on the same day. In fact, on the platform 33, the peak of activities is on the day 265

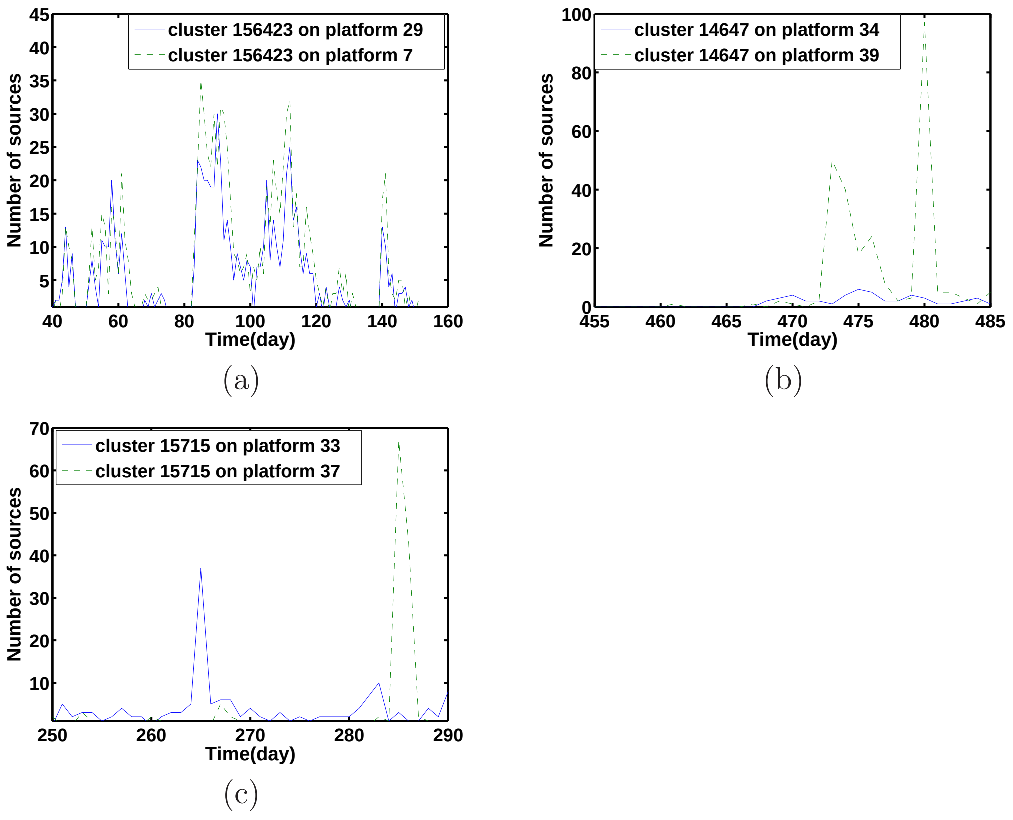


FIGURE 4.1 – Examples of evolutions of attack traces

whereas on the platform 37 the peak of activities is on the day 284. Here too, we do not expect the similarity measure to conclude that these two attack traces are correlated.

4.4 Measures

We present in this Section several functions to measure the distance (or similarity) of time series data. More precisely, the first three subsections present three techniques for measuring the similarity, and the second three subsections aim at representing three distance functions.

4.4.1 ScatterPlot

4.4.1.1 Description

A scatterplot is a useful summary of a set of bivariate data (two variables X and Y), usually drawn before working out a linear correlation coefficient or fitting a regression line [34]. It gives a good visual picture of the relationship between the two variables, and helps in the interpretation of the correlation coefficient or regression model. Each unit contributes one point to the scatterplot, on which points are plotted but not joined. The resulting pattern indicates the type and strength of the relationship between the two variables.

Here are some rules to recognize the correlation from a scatterplot ([34]).

- The more the points tend to cluster around a straight line, the stronger the linear relationship between the two variables (the higher the correlation).
- If the line around which the points tends to cluster runs from lower left to upper right, the relationship between the two variables is positive (direct).
- If the line around which the points tends to cluster runs from upper left to lower right, the relationship between the two variables is negative (inverse).
- If there exists a random scatter of points, there is no relationship between the two variables (very low or zero correlation).

Figure 4.2a shows the scatter plot of two random time series X and Y. In this example, both X and Y have the size of 100 and each element takes the random value from 0 to 100, as we can see, the points are randomly distributed. Figure 4.2b shows the scatter plot of two linearly correlated variable X and Y (The points tend to cluster around a straight line). And the positive slope of the line indicates that a large value of X corresponds to a large value of Y, and a small value of X corresponds to a small value of Y.

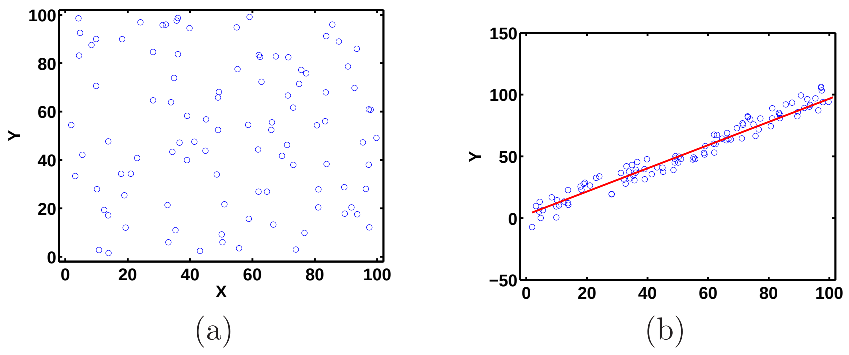
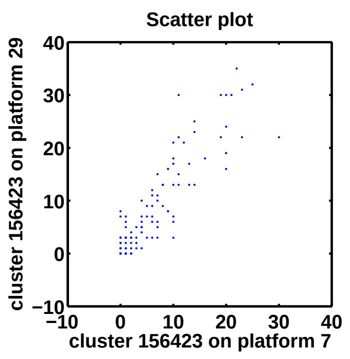


FIGURE 4.2 – a) A random distribution of points in the scatterplot indicates a no-correlation of X and Y b)Linear correlation (points tend to cluster around a straight line)

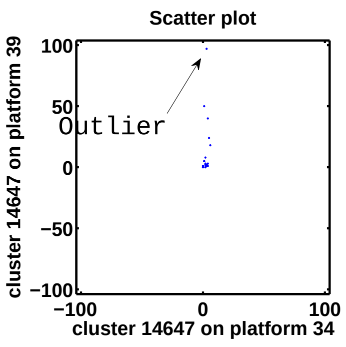
As an example, Scatter plots of attack traces represented in Figures 4.1a, 4.1b, and 4.1c are represented in Figures 4.3a, 4.3b, and 4.3c, respectively. Figure 4.3a shows that attack traces on Figure 4.1a are somehow correlated. Whereas, the outliers in scatter plots of Figures 4.3b and 4.3c shows that the important values of two activities in Figure 4.1b and 4.1c are not synchronized.

4.4.1.2 Discussion

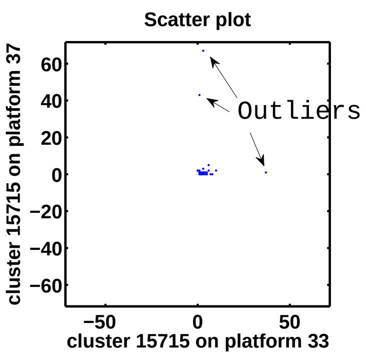
Scatterplot is a visualisation based method, it is intuitive and does not require special characteristic from data input. However, discovering the correlation of a large dataset is a time costly task as human participation is required. As clearly shown in Figure 4.3a, interpretation of a strong correlation is not always straightforward for



(a)



(b)



(c)

FIGURE 4.3 – Scatter Plots

many time series.

4.4.2 Pearson

4.4.2.1 Description

Pearson Product Moment Correlation Coefficient (Pearson correlation for short) is a correlation measure.The formula to compute the correlation coefficient between $X = (X_1, X_2, ...X_n)$ and $Y = (Y_1, Y_2, ...Y_n)$ is defined as follows :

$$r = \frac{1}{n-1} \sum_{i=1}^{n} (\frac{X_i - \overline{X}}{S_X})(\frac{Y_i - \overline{Y}}{S_Y}) \tag{4.2}$$

where $\overline{X}$ (resp. $\overline{Y}$) and S_X(resp. S_Y) are the mean and standard deviation of X (resp. Y). The product moment part of the name comes from the way in which the correlation coefficient is calculated : by summing up the products of the deviations of the data points from the mean.

4.4.2.2 Threshold

Pearson correlation is a similarity measure. It ranges from -1 to +1. A correlation of +1 means that there is a perfect positive linear relationship between time series. In other words, whenever X increases (or decreases), so does Y. A correlation of -1 means that there is a perfect negative linear relationship between variables. In other words, a high (low) value of X is associated with a low (high) value of Y. The level of correlation is usually based on wide-accepted thresholds described as follows [31] :

- -1.0 to -0.8 strong negative association.
- -0.8 to -0.5 moderate negative association.
- -0.5 to +0.5 weak or no association.
- +0.5 to +0.8 moderate positive association.
- +0.8 to +1.0 strong positive association.

As an example, the correlation coefficients of attack traces given in Figures 4.1a, 4.1b, and 4.1c are 0.91, 0.37, and -0.06, respectively. In other words, Pearson correlation confirms that only the attack traces in Figure 4.1a are correlated.

4.4.2.3 Discussion

Pearson correlation is able to compute the correlation even in the case of time series with differences in their scale. Figure 4.4 shows the evolution of two time series X and Y. The time series X experiences an abrupt change at point 21 whereas the time series Y is just slightly increasing. Despite the differences in amplitude, Pearson correlation gives a strong coefficient, 0.81. Depending on the application domain, we can consider this as either a strength or a weakness. We show how to benefit from this characteristic when working on aggregated attack traces in the next chapter.

Note that, Pearson's correlation coefficient only measures linear relationships between variables, and if data contains outlier, it will be greatly affected, since the value of $X_i - \overline{X}$ will become important in such a case.

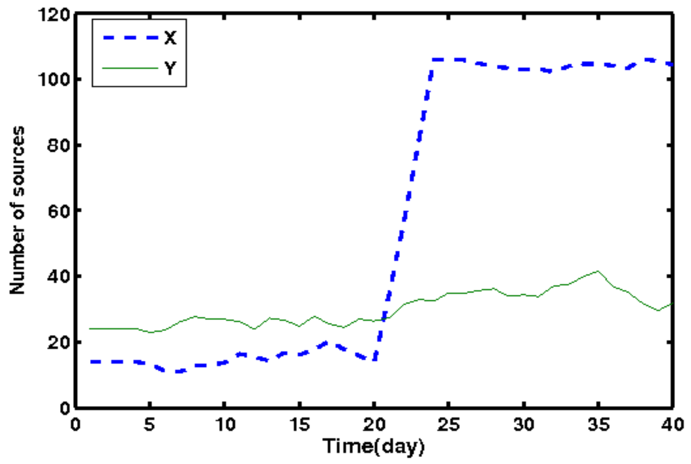


FIGURE 4.4 – Pearson correlation tolerates the difference in scale of time series data

4.4.3 Spearman

4.4.3.1 Description

It is sometimes important to know that one element is greater than the other one, no matter how different they are. In scale measurement, people call it ordinal scale. For instance, in a marathon race, the first runner reaching the target is ranked first, second comer is second, etc. no matter what the inter-arrival time between them is. Instead of comparing the real values, we can also compare objects by their ranks. Two time series will be seen as highly correlated if and only if their highest (resp. second, third, etc.) values show up in the same positions, no matter what values these points have. To calculate such correlation coefficient, Spearman correlation first transforms time series data to their rank-form and then applies Pearson correlation on the two rank-form time series. This is used for time series where the order is more important than the values themselves. To convert the time series $X = (X_1, X_2, ...X_n)$ into the rank-form $Y = (Y_1, Y_2, ...Y_n)$, we proceed as follows :

- Step 1 : X' is a sorted-form of X.
- Step 2 : We build $M' = ((X'_1, 1), (X'_2, 2), (X'_3, 3), ...(X'_n, n))$
- Step 3 : We build the final mapping table $M = ((DX_1, Y_1), (DX_2, Y_2), ...(DX_N, Y_N))$, in which $(DX_1, DX_2, ...DX_N)$ is the set of distinct values of X and Y_k is the mean of all Y'_j of all couples $(X'_j, Y'_j) \in M'$ which has $X'_j = DX_k$.

For illustrative purpose, given the time series $X = (6, 2, 7, 3, 30, 4, 3)$, the sorted-form of X is $X' = (2, 3, 3, 4, 6, 7, 30)$, and M' is ((2,1),(3, 2), (3,3), (4,4), (6,5), (7,6), (30,7)). The final mapping table is ((2,1), (3,2.5), (4,4), (6,5), (7,6), (30,7)). Thus the rank form of X is $Y = (5, 1, 6, 2.5, 7, 4, 2.5)$.

4.4.3.2 Threshold

As Spearman correlation applies Pearson correlation technique, the significance of the correlation coefficients should be interpreted in the same way as in 4.4.2.

As an example, the Spearman correlation coefficients of attack traces represented in Figures 4.1a, 4.1b, and 4.1c are 0.88, 0.7, and -0.09, respectively. Spearman correlation flags the first two examples as correlated.

4.4.3.3 Discussion

By working on ordinal data, Spearman correlation can avoid the problems caused by outliers in data. In fact, in the rank-form, the difference between outlier points and other data points will be significantly reduced, therefore, their impact will be less important in the rank-form than in the original values. For instance, the reason for which Spearman gives a high correlation value (0.7) for the attack traces represented in the Figure 4.1b is explained by the ranked forms of the two corresponding time series as represented in Figure 4.5.

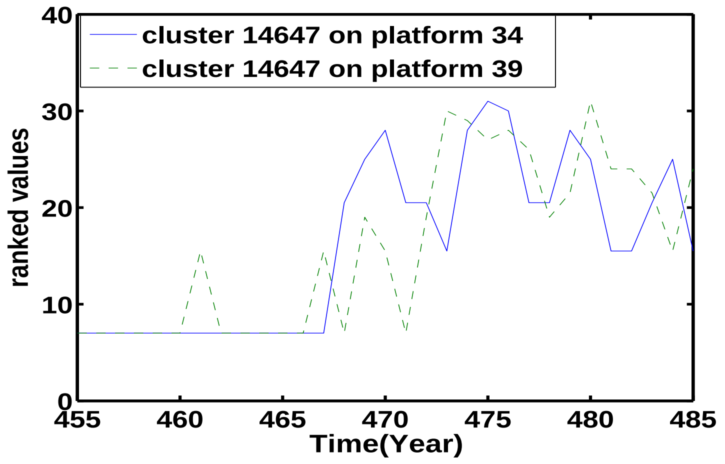


FIGURE 4.5 – The ranked values of attack traces from Figure 4.1b

4.4.4 Symbolic Aggregate Approximation

4.4.4.1 Description

Lin et al have proposed SAX (Symbolic Aggregate approXimation) in [70]. SAX allows to reduce the dimensionality of the time series. Simply speaking, to reduce the time series X of length n to w, SAX divides X into w equal frames to obtain the time series P. Each point in the newly obtained time series P is the mean of data points in the corresponding frame on the time series X. This technique is well-known under the name PAA (Piecewise Aggregation Approximations). The second step is to quantize the value of the newly created time series P into $\overline{P}$, called the SAX representation of P. More formally, SAX transformation is described as follows :

1. **Dimensionality Reduction Via PAA :** Time series $X = (X_1, X_2, ...X_n)$ is first normalized to obtain $X' = (X'_1, X'_2, ...X'_n)$ having a mean of zero and a standard deviation of one.
$$X'_i = \frac{(X_i - \overline{X})}{S_X} \tag{4.3}$$
(Where $\overline{X}$ and S_X are the mean and standard deviation of X, respectively). The normalized time series X' is then converted into its PAA representation $P = (P_1, P_2, ...P_w)$. Given the new time series length w, the i^{th} element of P is calculated with the following equation :
$$P_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} X'_j \tag{4.4}$$
(Note that $\frac{n}{w}$ is called the compression ratio).
2. **Discrimination :** The second step of SAX is to transform the PAA representation P of time series X into a discrete representation $\overline{P}$ where each value $\overline{P_i}$ is represented by a symbol belonging to a given alphabet. It has been shown that the best way to do the discrimination is to choose the quantization in such a way that every symbol has equiprobability distribution [7]. And since the normalized time series has the Gaussian Distribution by design [64], Lin et al have proposed to use the "break points" that will produce the equal size area under the Gaussian curve [64]. For example, the breakpoints for an alphabet size of 4 {a,b,c,d} are -0.67, 0, 0.67. In other words, it means that all the values of $P \leq -0.67$ will be represented by “a”, all the values of $P > -0.67$ and $P \leq 0$ will be represented by “b”... Figure 4.6 is an example of the output of this step.
3. **Distance :** SAX estimates the distance between two time series X and Y of length n as the distance between their SAX representations $\overline{PX}$ and $\overline{PY}$. For the sake of efficiency, inter symbol distances can be pre-computed and loaded into a lookup table TAB. For instance, when alphabet size $\alpha = 4$, the corresponding look up table is represented in Table 4.1. In this case, the distance between “a” and “b” is TAB(“a”,“b”)=0, between “a” and “c” is TAB(“a”,“c”)=0.67,

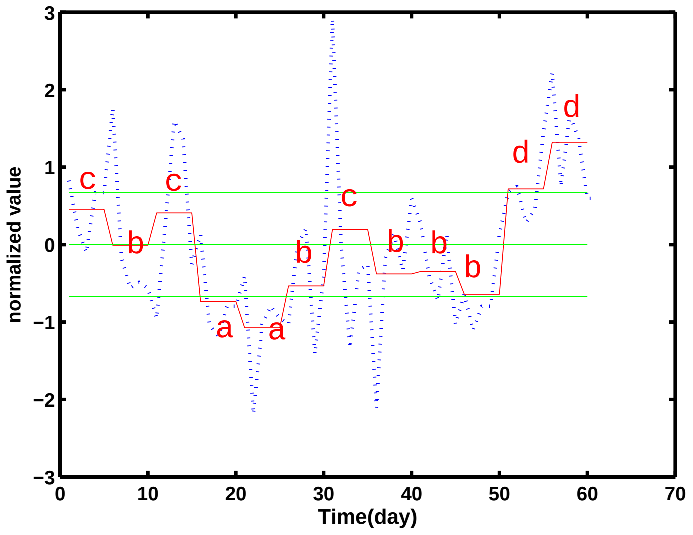


FIGURE 4.6 – Discrimination example

	a	b	c	d
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

TABLE 4.1 – Lookup table for alphabet α of 4

etc. Based on this, we define the distance between $\overline{PX}$ and $\overline{PY}$ as follows :

$$D(\overline{PX}, \overline{PY}) = \sqrt{\frac{n}{w} \sum_{i=1}^{w} TAB(\overline{PX}(i), \overline{PY}(i))} \tag{4.5}$$

where $\overline{PX}(i)$ and $\overline{PY}(i)$ are the i-th symbols of $\overline{PX}$ and $\overline{PY}$, respectively.

4.4.4.2 Threshold

SAX is a distance measure. Unlike the Pearson and Spearman measures, there is no fixed threshold to determine the similitude of two time series. In fact, the distance value depends on many factors : compression ratio, alphabet size...

As an example, when applying SAX distance with different alphabet sizes, and the compression ratio of 1 to the attack traces represented in Figures 4.1a, 4.1b, and 4.1c, we obtain the distances given in Table 4.2.

4.4.4.3 Discussion

SAX is a flexible and powerful tool for dimensionality reduction. It is flexible since we can adjust the compression ratio and the alphabet size. But of course, this

α	Figure 4.1a	Figure 4.1b	Figure 4.1c
4	1.34	1.5	1.5
5	0.71	2.31	1.65
6	1.18	1.79	2.27

TABLE 4.2 – Examples of SAX distances

is a trade off. In Figure 4.7, time series Y has a slight peak at point 13 and a strong peak at point 20 whereas time series X has a strong peak at point 13 and a slight peak at point 20. Applying SAX with $\alpha = 5$ and $w = 30$ (the same as the original length of the original time series), we obtain the two alphabet strings $\overline{CX}$ and $\overline{CY}$ as follows :

$\overline{PX}$=2 3 3 3 3 3 3 2 3 2 2 3 **4** 3 2 3 3 2 2 **5** 2 3 3 3 3 2 3 2 3 3

$\overline{PY}$=2 2 3 3 2 3 3 3 3 3 2 3 **5** 3 3 3 3 3 3 **4** 3 2 3 3 2 3 3 2 2 3

SAX returns a distance of zero for these two time series which corresponds to the highest level of similarity that two time series can obtain. It is difficult to judge whether it is good or bad without considering this in a concrete application context.

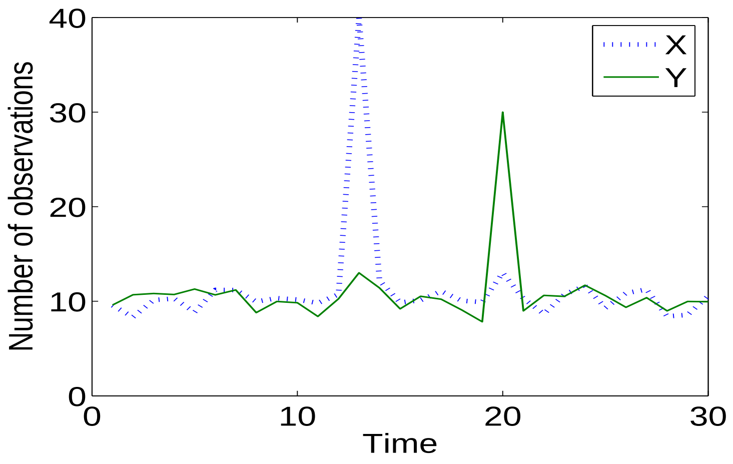


FIGURE 4.7 – SAX : distance of zero

4.4.5 Minkowski

4.4.5.1 Description

Another important distance measure is Minkowski. Given two time series $X(X_1, X_2, .., X_n)$ and $Y(Y_1, Y_2, .., Y_n)$, the Minkowski distance of order p (p-norm distance) is defined as follows :

$$D = \left(\sum_{i=1}^{n} |X_i - Y_i|^p \right)^{\frac{1}{p}} \tag{4.6}$$

In practice, the most frequently chosen values for p are :

p	Figure 4.1a	Figure 4.1b	Figure 4.1c
2	49.7	114.7	86.9
3	28.9	100.1	72.7
4	23.2	96.2	68.2

TABLE 4.3 – Examples of Minkowski distances

1. **p=1, Manhattan :** The Manhattan distance is also known as City Block distance. This function computes the distance that would be traveled to get from one data point to the other if a grid-like path is followed. The Manhattan distance between two items is the sum of the differences of their corresponding components. The formula for the distance between $X = (X_1, X_2, ...X_n)$ and $Y = (Y_1, Y_2, ...Y_n)$ is :

$$D = \sum_{i=1}^{n} |X_i - Y_i| \tag{4.7}$$

2. **p=2, Euclidean Distance :** The Euclidean distance function measures the *as-the-crow-flies* distance. The formula for such distance between $X = (X_1, X_2, ...X_n)$ and $Y = (Y_1, Y_2, ...Y_n)$ is :

$$D = \sqrt{\sum_{i=1}^{n} (X_i - Y_i)^2} \tag{4.8}$$

   According to [61], Euclidean distance is used in 80% cases as distance measure of time series.

3. $p \to \infty$**, Chebychev Distance** The Chebychev distance between X and Y is the maximum distance between the points in two time series.

$$\lim_{p\to\infty} \Big(\sum_{i=1}^{n} |X_i - Y_i|^p \Big)^{\frac{1}{p}} = max(|X_1 - Y_1|, |X_2 - Y_2|, ..., |X_n - Y_n|) \tag{4.9}$$

   The Chebychev distance may be appropriate if the difference between points is reflected more by differences in individual dimensions rather than all the dimensions considered together.

4.4.5.2 Threshold

Minkowski is a distance measure, the distance value depends on the nature of the data, the length of time series, and also the value of p. So, in order to determine that two time series are similar, we must fix the threshold manually.

Applying the Minkowski distance with different values of p to attack traces represented in Figure 4.1a, 4.1b, and 4.1c, we obtained the distances as represented in Table 4.3.

4.4.5.3 Discussion

The Minkowski distance is based on the sum of the differences $|X_i - Y_i|^p$ (from Equation 4.6) of the corresponding elements in two time series. By changing the parameter p, we can adjust the impact of these differences on the final distance. Actually, the greater the value of p is, the less important the small differences are and when $p \to \infty$ the distance is equal to the maximum difference of $|X_i - Y_i|$ (Chebychev Distance). At the contrary, when p is small, the contribution of the small differences $|X_i - Y_i|$ to the total distance increases.

It is obvious that we can not apply this technique to raw data since it will introduce a large distance even in case of high correlation. As an example, Figure 4.8a represents two similar time series, but it has a large Minkowski distance due to the differences in amplitude (illustrated in Figure 4.8b).

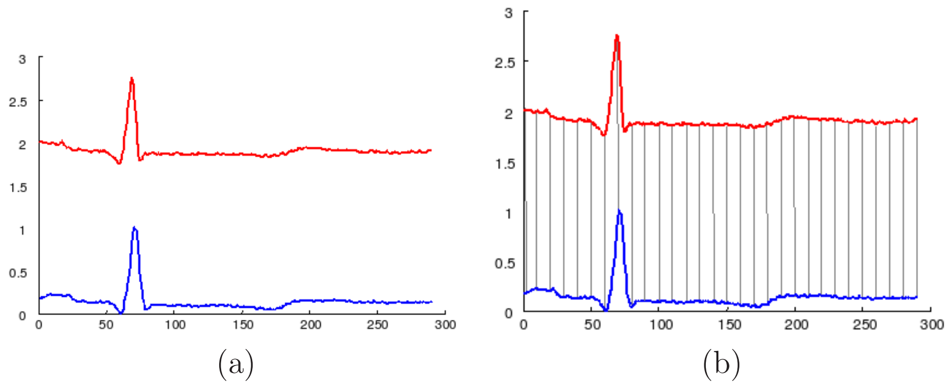


FIGURE 4.8 – a)Two similar time series (reprinted from [61]) b)Large Minkowski distance due to the difference in amplitude(reprinted from [61])

4.4.6 Dynamic Time Warping (DTW)

4.4.6.1 Description

DTW is introduced in [62] and mostly used in speech recognition thanks to its ability to treat the out of phase time series. In addition to speech recognition, DTW has also been found useful in many other disciplines [60], including data mining, gesture recognition, robotics, manufacturing, and medicine. Given the two time series $X = (X_1, X_2, ...X_M)$ and $Y = (Y_1, Y_2, ...Y_N)$, we construct a cost matrix M-by-N (illustrated on Figure 4.9) in which the value of cell (i,j) is the corresponding distance $d(X_i, Y_j)$ between X_i and Y_j (Euclidean distance is usually used, so $d(X_i, Y_j) = (X_i - Y_j)^2$). We denote each cell (i, j) a certain W_k. The warp path $W = (W_1, W_2, ...W_K)$ is the list of cells satisfying the follow conditions :

1. The warp path's length must be greater than or equal to the maximal length of two time series and smaller than the sum of both of them $MAX(M, N) \leq K \leq M + N - 1$

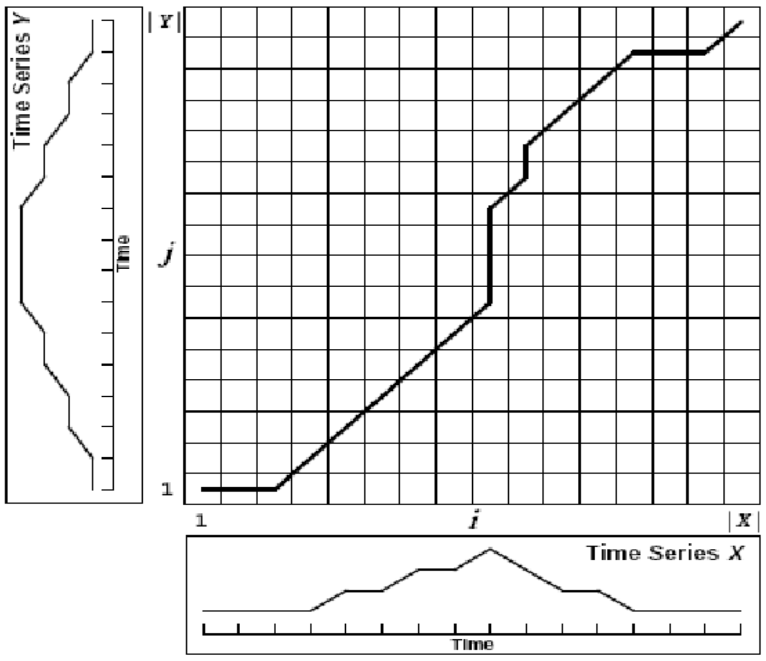


FIGURE 4.9 – A cost matrix with the minimum-distance warp path traced through it (reprinted from [104])

2. The warp path starts from the beginning of each time series $W_1 = (1, 1)$ and ends at each time series $W_k = (M, N)$
3. The index must be monotonically increasing : $W_k = (i, j), W_{k+1} = (i', j') \quad i \leq i' \leq i + 1, j \leq j' \leq j + 1$.

The DTW distance is the minimum-distance warp path, which is defined as follows :

$$DTW(X, Y) = \sum_{k=1}^{K} \overline{W_k} \tag{4.10}$$

where $\overline{W_k}$ is the value of the corresponding cell W_k of the matrix.

4.4.6.2 Threshold

DTW is a distance measure. Similarly to the SAX and Minkowski distances, the threshold must be fixed manually. The distances of attack traces represented in Figure 4.1a, 4.1b, and 4.1c are 31.4, 109.1, and 5.38, respectively.

4.4.6.3 Discussion

DTW, like the Minkowski distance, also suffers from the scaling issue. Therefore, the time series data need to be normalized before applying DTW. Besides that, the out-of-phase feature sometimes introduces unexpected results. For instance, considering attack traces of cluster 15715 on platforms 33 and 37 as represented in Figure 4.1c, each of them contains only one important peak of activities but they are out-of-phase. It is obvious that, without further analysis, there is no means for us to conclude that these two activities are linked together. It is why we do not expect that the correlation measures flag them as correlated. However, DTW concludes that their distance is 5.38 which is much smaller than the value for Figure 4.1a where curves are clearly correlated.

4.5 Discussion

4.5.1 Measure Requirements

We have presented so far several techniques that can be used to compute the distance or similarity between two time series data. Before deciding which one best suits our needs, we go to some more levels of detail about our application context. First, regarding the operational requirement, we need a technique that is applicable to a large dataset (as described in Chapter 3). Second, concerning the functional requirement, in the context of malicious network traffic, if the time series is stable, we consider that there is no special activity happening. Otherwise, if time series exhibits variation or peaks of activities, we consider that something different is happening, worthwhile being identified and analyzed.

With respect to the application context as described above, we identify the following mandatory properties for the ideal similarity function $\mathscr{P}()$

- **Synchronization :** $\mathscr{P}()$ *should not return true if peaks of activities are not synchronized.* The reason for this is that the important activities are the most interesting ones in the time series. If a $\mathscr{P}()$ concludes that the two time series are correlated, their important activities must be synchronized.
- **Scalability :** $\mathscr{P}()$ *must be applicable to a large dataset.*

Besides that, from the practical viewpoints, we propose two other supplementary requirements. These are not mandatory requirements to exclude a technique, but they have some impact on the final choice.

- **Threshold determination :** One of the basic tasks when comparing objects is to determine the threshold. With the same magnitude, two time series are, are not, correlated on the basis of such value. For some correlation functions, we have well-accepted thresholds, for some others, the threshold needs to be fixed manually. For instance, SAX distance function is influenced by the alphabet size, and the compression ratio. Clearly, having well defined thresholds greatly helps in automatizing the approach. To simplify things, *we plead for the correlation function $\mathscr{P}()$ having some wide-accepted predefined threshold.*
- **Pre-processing requirement :** Some correlation functions require the pre-processing step, i.e, normalization. We tend to choose the correlation function $\mathscr{P}()$ that does not require the pre-processing task if there are two equal candidates, for cost reason.

4.5.2 Discussion

This Section provides the discussion about the applicability of different correlation functions with respect to our application context. All techniques are evaluated with respect to the level of satisfaction of the requirements mentioned in 4.5.1. As far as the *synchronisation* criteria is concerned, SAX does not ensure synchronization of important activities because of the data reduction step, as shown in Figure 4.7. Spearman can reduce the impact of outliers, but in our application context, the outliers correspond to important activities that characterize the cluster time series. In other words, Spearman correlation's strength becomes its weakness in our case. DTW has the capacity of treating out-of-phrase time series, whereas, in our context, the important activities must be synchronised. Again the capacity of treating the out-of-phase time series has a negative impact in our context. Regarding the *scalability requirement*, only Scatterplot is disqualified, since it is a visualization based approach, it needs the participation of human for the judgement. As discussed earlier in Section 4.4.5 and Section 4.4.6, DTW and Minkowski distances require the pre-processing step to normalize data input. Finally, regarding the *threshold determination* criteria, the significance of correlation coefficient given by Spearman and Pearson are widely accepted, whereas in the case of SAX, Minkowski, and DTW distance, we must fix the threshold manually.

Table 4.4 is the summary of our discussion. Based on this, Pearson appears to be the only technique satisfying all our requirements and is, therefore, chosen in the sequel of this work.

	Measure	Synchro -nisation	Scalability	Threshold determination	Preprocessing
Scatter	C	Yes	No	NA	No
Spearman	C	No	Yes	Fixed	No
SAX	D	No	Yes	Manual	No
DTW	D	No	Yes	Manual	Yes
Minkowski	D	Yes	Yes	Manual	Yes
Pearson	C	Yes	Yes	Fixed	No

D :*Distance*, **C** :*Correlation*, **NA** :*Non-Applicable*

Table 4.4 – Comparison of correlation techniques

4.6 Summary

We have introduced several techniques to compute the similarity of time series data. For each technique, we have studied in detail its mathematical properties. In the context of comparing malicious network traffic with a high volume of data, we have proposed four criteria coming into play for the choice of the appropriate correlation technique. We have concluded that Pearson correlation is the most suitable for our case.

x

Chapter 5

ON THE IDENTIFICATION OF ATTACK EVENTS

5.1 Introduction

As said earlier, we want to detect the macro attack events, that is, the groups of correlated micro attack events. For this to happen, we need first to choose the best suited correlation techniques for our application domain. This issue has been discussed in Chapter 4. Second, given a large dataset (as presented in Chapter 3) we need efficient methods to find the similarities between all the attack traces at our disposal. In this Chapter, we explain why we need all of that in order to identify the micro and macro attack events. This is justified when we discuss the two approaches we use to detect the micro and macro attack events. In the first approach, to detect the macro attack events, we need first to detect the micro attack events. At the contrary, in the second approach, we start with the identification of macro attack events to subsequently find the micro attack events. The remainder of this Chapter is organized as follows. Section 5.2 shows how we reduce the number of attack traces to be considered, thus, to reduce the computational cost. In Section 5.3, we describe the first approach, i.e. to detect first the micro attack events, and, then, to identify macro attack events. In Section 5.4, we present the second approach, i.e. to detect first the macro attack events, and, then, the micro attack events. Section 5.5 presents the experimental validation of our approach. Finally, Section 5.6 concludes the Chapter.

5.2 Pre-processing

Before going to the detailed discussion about the pre-processing technique, we introduce a very important concept, **observed cluster time series**, in the following.

Definition 8 ***An Observed Cluster Time Series*** *$\Phi_{T,c,op}$ is a function defined over a period of time T, T being defined as a time interval (in days). That function* *returns the amount of sources per day associated to the cluster c*[1] *as seen from the given observation viewpoint op. The observation viewpoint op can either be a specific platform or a specific country of origin. In the first case,* $\Phi_{T,c,platform_X}$ *returns, per day, the amount of sources belonging to cluster c that have hit* $platform_X$*. Similarly, in the second case,* $\Phi_{T,c,country_X}$ *returns, per day, the amount of sources belonging to cluster c that are geographically located in* $country_X$*. Clearly, we always have :* $\sum^{\forall i \in countries} \Phi_{T,c,i} = \sum^{\forall x \in platforms} \Phi_{T,c,x}$

As indicated later, the computational overhead problem comes from the fact that we have too many observed cluster time series. In fact, to learn whether an observed cluster time series is correlated with another observed cluster time series, we must compare that observed cluster time series to all the others. It is evident that the more observed cluster time series we have, the more expensive the computational effort is. As an indication, we have around 400,000 observed cluster time series as of now. Such a high number of observed cluster time series is the main reason for computational overhead. To deal with this, we start with the following observation. Let us assume that there is an easy way to put the observed cluster time series into different classes, we can distinguish two kinds of correlation as follows :

- **within-class correlation :** it is to compute the correlation of observed cluster time series belonging to one class.
- **cross-class correlation :** it is to compute the correlation of observed cluster time series belonging to different classes.

If, by design, the classes we have built are such that there should be no significant correlation between two observed cluster time series belonging to two different classes, then we will dramatically decrease the computational effort by limiting ourselves to the evaluation of the “within class correlation”. In the following, we discuss how to build these classes.

5.2.1 Assumption

Our assumption is that the observed cluster time series could be classified into different categories based on their level of stability. The observed cluster time series of certain attack tools may vary more than the others. There may be many reasons for that to happen. For instance, several attack processes may exploit the attack tools differently. Or it may be due to the varying of the amount of infected machines (new machines are infected, others get patched). With that in mind, we make the assumption that the observed cluster time series can be classified into the following three categories :

1. **Peaked family** : observed cluster time series in this family exhibit a significant peak of values during a very small period of one or two days and almost no activity otherwise. The fact that several sources send packets simultaneously to a single platform in a short period of time shows the highly coordinated nature of the attacking machines.

1. The notation of cluster, as used here, is the one defined in Chapter 3, Section 3.3.2.2

2. **Stable family** : observed cluster time series in this family have a roughly constant behavior during their whole lifetime.
3. **Strongly varying family** : observed cluster time series in this family are characterized by wide amplitude variations over, possibly very, long periods of time.

There are many advantages with the above classification. Firstly, we do not need to compute the *cross-class correlation* since the stable observed cluster time series can not be correlated neither with peaked observed cluster time series nor with strongly varying observed cluster time series. Similarly, the observed cluster time series in the strongly varying family are not likely to be correlated with peaked ones. Secondly, since computing the correlation of stable observed cluster time series does not bring much sense, we do not need to compute the *within-class correlation* for the observed cluster time series in this class.

5.2.2 Validation

To validate our above assumptions, for each observed cluster time series, we first identify its active period or lifetime. The lifetime of an observed cluster time series is defined as the period from the first day to the last day during which that observed cluster time series is observed. If the lifetime of one observed cluster time series is only one day, then, we declare that the observed cluster time series belongs to the peaked family. Otherwise, we compute the standard deviation of the observed cluster time series over its lifetime. If it is smaller than a threshold δ, then we flag the observed cluster time series as belonging to the stable family. Otherwise, we filter out the outlier data point from the time series to form the filtered time series. Outlier data point is defined as the maximum value of the time series. Then we compute the standard deviation of filtered time series. If the standard deviation is now smaller than δ, we declare the time series as being a peaked time series[2]. Otherwise, we declare the observed cluster time series as belonging to the strongly varying family.

Figure 5.1 illustrates the algorithm for an observed cluster time series that spans over 20 days. The standard deviation of the time series is 6.51. Since it is greater than 1, our algorithm does not declare this time series as a stable one. We next filter the extreme values from this time series. This boils down to cutting the peak on day 12. The resulting time series is obviously smoother than the initial one and its standard deviation is 0.46, which is smaller than the threshold 1. Hence, our algorithm eventually flags the time series of Figure 5.1 as belonging to the peaked family.

5.2.2.1 Threshold

We test the approach with a subset 20,756 observed cluster time series extracted from a bigger dataset as presented in Section 5.5. As mentioned earlier, our intention

2. It is different from the first case in the sense that its lifetime is greater than 1, and it has the stable behavior during its whole lifetime besides on a single point

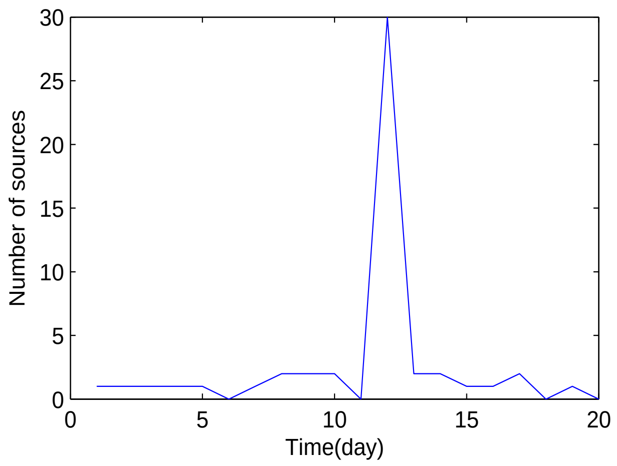


Figure 5.1 – Example of the peaked family time series

is to classify them into three different categories. The problem now is to choose a good threshold δ to separate them. Suppose that there exists an ideal threshold δ' that can classify correctly observed cluster time series into three different classes. For $\delta \leq \delta'$, we may misclassify a stable time series as a peaked or strongly varying time series. In this case, we will spend more computational effort than we could have with δ', but we will never miss any correlation (i.e., we will never conclude that two varying time series are not similar whereas they are). For $\delta \geq \delta'$, we could consider the strongly time series or peaked time series as stable. In this case, there are less computational effort but we may not detect all the phenomena existing in the dataset. For the sake of completeness, we tend to choose $\delta \leq \delta'$. Of course we can not visually inspect all the time series to choose the good threshold δ. We adopt the following heuristic approach, we first choose a random value of $\delta = v$, then we extract the set of time series identified as strongly varying or peaked time series for $\delta = v$, but not when $\delta = v + \epsilon$. If the visualization tells us that these observed cluster time series are all stable time series, we know that $\delta \leq \delta'$, we increase δ by ϵ. Otherwise, we decrease δ by ϵ. We stop the process when δ is stable.

We have chosen $\epsilon = 0.2$, and by applying the heuristic described as above, we end up with $\delta = 1$. It is important to notice that we have tested with smaller value of ϵ and it has been shown that the smaller granularity of ϵ has very small impact on the final result. Figure 5.2a shows the evolution of three classes (For the visibility, Figure 5.2b represents the distribution of only peaked and strongly varying time series). The figure tells us that most of time series fall within the stable time series class. Concretely for $\delta = 1$, we have 19564 time series in stable family, 375 time series in the peaked family and 835 time series in the strongly varying family. This result shows the usefulness of the pre-processing step, since most of the cluster time series fall within the stable family for which we do not need to compute neither the cross-classes correlation nor the within-class correlation.

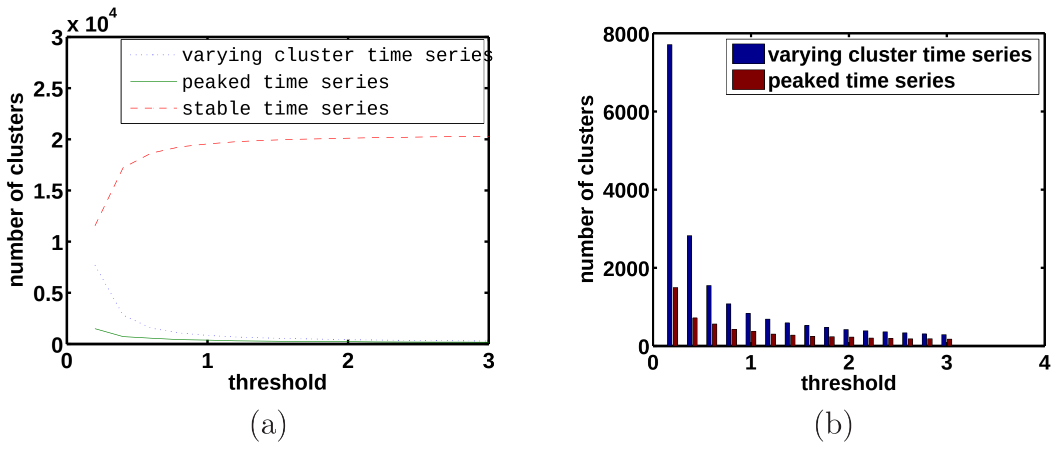


FIGURE 5.2 – Plot a represents the evolution of size of three classes over threshold(Plot b represents a two of them for the visibility)

5.3 Micro Attack Event Approach

To detect the micro and macro attack events, this approach starts first by identifying individually the micro attack events, and then based on that, detects the macro attack events.

5.3.1 Micro Attack Event Detection

A micro attack event, as defined in Chapter 1, is made of a set of attacking machines, having left the same attack fingerprint, observed over a limited period of time. Our assumption is that attacking sources that are part of the same attack campaign will have a special distribution both in terms of time and space. In our context, we have represented the attack traces by the observed cluster time series. That is the evolution of the use of a given attack tool seen in one place. In this sense, if attacks of one cluster targeting one place suddenly increase and stop, they should be considered linked to each other. In other words, this would be considered as a micro attack event. We classify them into two classes.

- A set of such micro attack events is detected in the pre-processing step, and they are considered as peaked observed cluster time series. As an example, Figure 5.3 represents an observed cluster time series from the peaked family. It consists of activities generated by cluster 149315 happening on day 55 on platform 44. These activities are caused by around two hundreds sources. Note that, we only keep peaks at least 50 sources. This is supposed to be a conservative threshold to eliminate the background noise.
- Micro attack events may also exist as peaks in the strongly varying observed cluster time series. As an example, Figure 5.4 represents the evolution of cluster 14647 on platform 1 over a period of 150 days. As we can notice, there are three peaks of activities on three different days. Our goal is to have those peaks identified as micro attack events by our approach. To detect them, we proceed

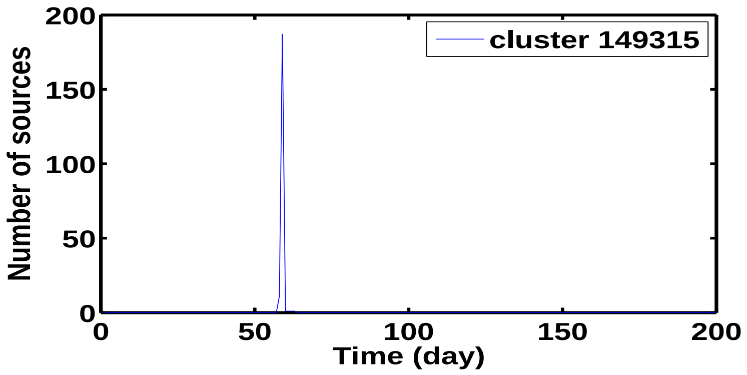


Figure 5.3 – Peaked cluster time series 149315 on platform 44

as follows. Since the outliers are locally compared to their neighbourhood data items, we first split a long time series Φ into several sub time series s of smaller size. For each sub time series s, we then need to verify if there are outliers in it. Since the values of the outliers, if any, are much more important than those of the others, the standard deviations of time series s with/and without these outliers must be very different. More precisely, on each time series, we remove an amount of ten percent of the number of data points in s to form a filtered time series. The removed items are the most important ones. In our case, we express the difference between two standard deviations by their ratio. If this ratio exceeds a given threshold η, we know that there are the peaks of activities in the sub time series. In this analysis, the length of s is 30. We have tested different values η and experiment shows that value η of 3 gives good results. If the above process concludes that sub time series s indeed contains the outliers, we proceed as follow to detect them. To identify the peaks, we compare the data points of s with a given threshold δ. Any data points greater than δ are considered as peaks (the method is well-known under the name Peak Picking algorithm). Our problem now lies in how to choose δ. One possible solution is based on the average of the population. For instance, the author of [87] has proposed to use the threshold δ as twice the average of population. Experiment shows that it works well especially when the peaks expand in a short time interval. This happens to be the situation we are in.
Applying the Peak Picking algorithm represented in Alg. 1 to the example in Figure 5.4, we obtain three micro attack events on three distinct time intervals [485,485], [561,561], and [575,575], i.e. exactly what we wanted.

5.3.2 Detection of Macro Attack Events

Since the micro attack events detected by the previous method have short life spans (one or two days), we consider that any other micro attack event happening in the same period of time should be seen as being correlated. To build the macro attack events, we just need to group together all the micro attack events happening on the same time interval. As an example, Figure 5.5 represents the evolution of cluster

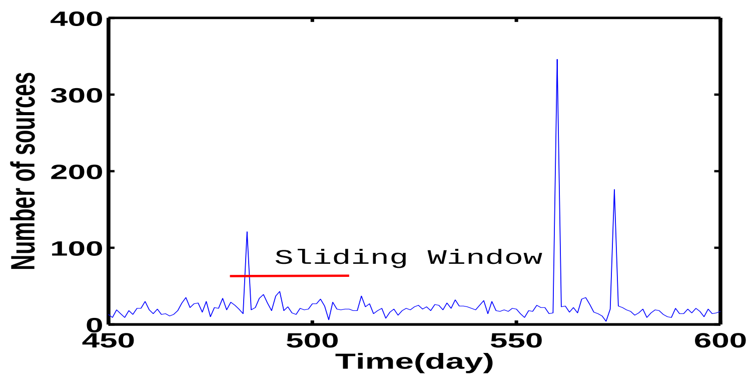


FIGURE 5.4 – Three micro attack events of cluster 14647 observed on platform 1

Alg. 1 : Peak Peaking Function

```
Input :
        $X = (X_1, X_2, ...X_n)$ : Time Series
Output :
        $I_X$ Index of peaks
begin
  1     choose the right $\delta$
        for i=1 to n
  2         if $X_i > \delta$
  3             $I_X(i) \leftarrow 1$
  4         else
  5             $I_X(i) \leftarrow 0$
  6         end
        end
end
```

0 on platform 1. Applying the Peak Picking technique, we obtain the three similar time intervals [485,485], [561,561], and [575,575] with the ones detected earlier in Figure 5.4. Combining with the previous example in Figure 5.4, we obtain three macro attack events. They all consist of two cluster 0 and 14647 on platform 1, but happen on three distinct time intervals [485,485], [561,561], and [575,575].

We denote the (micro and macro) attack event i as $e_i = (T_{start}, T_{end}, S_i)$ where the (micro and macro) attack event starts at T_{start}, ends at T_{end} and S_i contains a set of observed cluster time series identifiers (c_i, op_i). If S_i is a singleton set, e_i is a micro attack event. Otherwise, e_i is a macro attack event, and all $\Phi_{[T_{star}-T_{end}],c_i,op_i}$ are strongly correlated to each other $\forall(c_i, op_i) \in S_i$.

Applying to the previous case, we have three macro attack events : $e_1 = (485, 485, \{(14647, 1), (0, 1)\})$, $e_2 = (561, 561, \{(14647, 1), (0, 1)\})$, and $e_3 = (575, 575, \{(14647, 1), (0, 1)\})$.

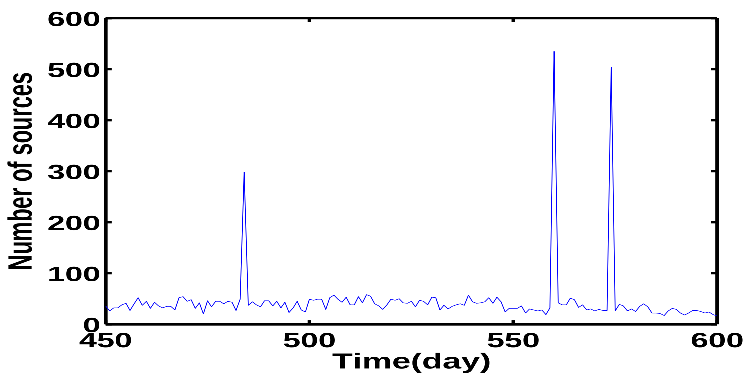


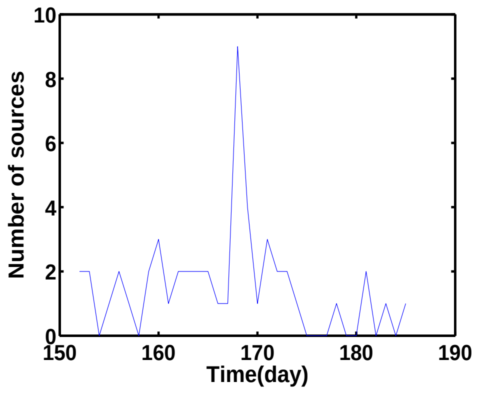
FIGURE 5.5 – Three micros attack events of cluster 0 observed on platform 1

5.4 Macro Attack Event Approach

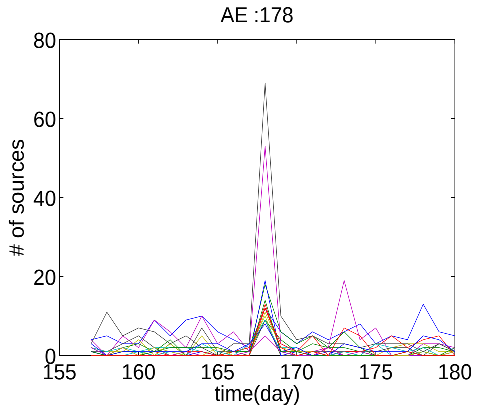
5.4.1 Introduction

Applying the approach as presented in Section 5.3, we can identify phenomena that are locally visible. In other words, the phenomena must have great impact on the observed cluster time series, so that, they are detectable. By doing so, we somehow do not leverage the strength of our distributed system, i.e., to observe the attacks at different places. In fact, there may exist phenomena that may not be detected by using local information, but are detectable when using global information. As an example, Figure 5.6a represents the evolution of attacks of the cluster 14647 on the platform 45. We see a small peak of activities on day 169 (9 sources only), but it is not a strong evidence indicating that something special is happening. Such event, if looked at it in isolation, will be considered to be part of the radiation noise that exists on the Internet. But when we look at the other observed cluster time series on this period of time, we identify 16 other observed cluster time series which also exhibit peaks of activities at the very same day. This suggests to use another approach that can correlate this kind of information to detect micro and macro attack events.

To detect macro attack events that involve several observed cluster time series having a small number of sources, we apply the time series correlation technique to extract the groups of correlated *observed cluster time series*, i.e., the macro attack events. More precisely, Section 5.4.2 represents what we call the correlation sliding window approach to detect the macro attack events. The approach takes the observed cluster time series as input and returns the macro attack events existing in the observed cluster time series. To make our approach applicable for different contexts, we develop in Section 5.4.3 two particular uses of the correlation sliding window method that can treat large datasets.



(a)



(b)

FIGURE 5.6 – a) evolution of cluster 14647 on platform 45. b) several correlated observed cluster time series from 10 platforms

5.4.2 Correlation Sliding Window

The algorithm consists of five steps. In the first step, we compute the correlation coefficients for each pair of observed cluster time series. In the second step, we identify the correlated time intervals for each pair of observed cluster time series. In steps 3 and 4, we identify what we call *common correlated time interval*, in which several time series are correlated. And in step 5, we identify the macro attack events on the *common correlated time interval* basis. We give the detailed description for each step in the following.

5.4.2.1 Step 1 : Correlation Sliding Window

Given two time series Φ and Ψ, of length T, we want to identify the correlated pair $P_i = \{start, stop, \Phi, \Psi\}$ where two time series Φ and Ψ evolve similarly from day *start* to day *stop*. To obtain these correlated time intervals, one of the approach consists in computing the correlation coefficient of all possible sub periods $[a, b]$ with $(0 \leq a \leq b \leq T)$, but this would be extremely costly. To avoid this, we make use of the correlation sliding window technique. The idea is that we compute the correlation coefficient of successive time windows (with a fixed length of L) on Φ and Ψ, moving from the left to the right. Let $cor(A, B)$ be the correlation coefficient of two vectors A and B. The correlation vector C of Φ and Ψ is computed as follows :

$$C[k] = cor(\Phi[k, k+L-1], \Psi[k, k+L-1]), \quad k = 1, \dots T - L + 1 \tag{5.1}$$

($cor(\Phi[k, k+L-1], \Psi[k, k+L-1]$) is the correlation coefficient of the time window from k to $k+L-1$)

As one can imagine, if the value of L is too high, we may not discover the short correlated period of two time series. Otherwise, if value of L is too small, the correlation value may be due to the random factor. In reality, we have tested different values of L, and found that L=30 gives good results.

5.4.2.2 Step 2 : Identification of Correlated Time Intervals of Two Time Series

Given a threshold δ and a correlation vector C of two time series Φ and Ψ, our goal in this step is to identify time intervals where these two time series are considered as being correlated. In other words, we try to identify the correlated pair $P_i = \{start, stop, \Phi, \Psi\}$. To this end, we have tested the following three techniques, namely *strict policy*, *tolerant policy*, and *tolerant policy with adjustment*. They are described as follows :

- **Tolerant policy :** We call C' the vector that holds the correlation status of Φ and Ψ. At first, we assign $C'(i) = 0, \quad \forall i \in [1, T]$. Then, for each $C[k] \geq \delta$, we update the status of C' in the time interval $[k, k+L-1]$ to 1, ($C'(i) = 1, \quad \forall i \in [k, k+L-1]$). Φ and Ψ are considered as being correlated in the time interval $[a, b]$ when $C'(i) = 1, \quad \forall i \in [a, b]$.
  An important parameter in our procedure is the choice of the threshold δ to declare that two time series are correlated. Again, we rely on experience, i.e., visual inspection of a lot of cases for different values of δ, to choose our threshold. We end up having a threshold of 0.7.
  Figure 5.7 represents the evolution of two observed cluster time series from day 1 to day 100. The dashed curve represents the correlation value (vector C), the threshold curve (dotted curve) intersects it at two points t1, and t2 (corresponding to X=9 and X=40, respectively). In this specific example, we use the sliding window of size 30. In the *tolerant policy*, we define these two time series as correlated from day 9 to day 69. From a statistical viewpoint, this is totally correct since the two time series are indeed similar, w.r.t the correlation value calculated, during this period. However, when looking at the network activity we see that this is due mostly (if not all) to the time interval from day 38 to day 40 during which most of the important synchronized activities happen.
- **Strict policy :** In contrast to the previous case, we first assign $C'(i) = 1, \quad \forall i \in [1, T]$. Then for each correlated value $C[k] < \delta$, we update the status of C' in the time interval $[k, k+L-1]$ to zero ($C'(i) = 0, \quad \forall i \in [k, k+L-1]$).

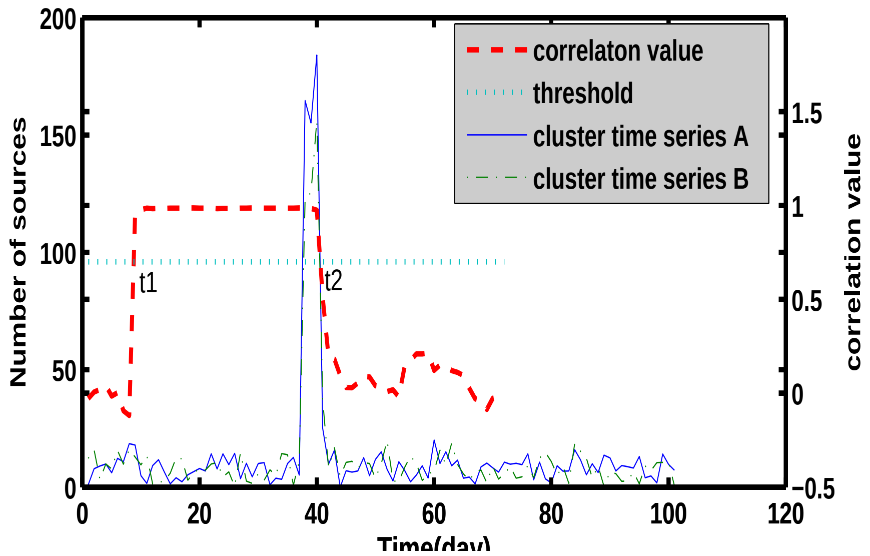


FIGURE 5.7 – The *tolerant policy* says that the two time series are correlated from day 9 to day 69 whereas the *strict policy* says from day 38 to day 40

Φ and Ψ are then considered as being correlated in the time interval $[a, b]$ when $C'(i) = 1 \quad \forall i \in [a, b]$.

Applying the *strict policy* to the same example in Figure 5.7, we obtain the correlated time interval from day 38 to day 40, which is much more precise than what is provided by the *tolerant policy*. However, the *strict policy* faces another kind of error. For instance, Figure 5.8 represents the same example as on Figure 5.7, except that there are peaks of activities from day 60 to day 63 on the sole time series A. Due to the interference of these activities, the correlation values degrade significantly. This time, the threshold curve (dashed curve) intersects the correlation value curve (dashed dotted) at two points t1(X=9), and t2(X=33). As a consequence, strict policy can not discover the correlated time interval from day 38 to day 40 (note that our sliding window's length is always equal to 30). We use the term "dismissal problem" to indicate this kind of error.

– **Tolerant policy with adjustment :** as discussed earlier, the tolerant policy suffers from some inaccuracy when detecting the correlated time interval and the strict policy faces the dismissal problem when identifying the correlated periods of two time series. We want a more robust technique to achieve this goal. Our solution is, first, to apply the tolerant policy to get the temporal correlated time intervals, then we apply the post-processing step to eliminate the potential inaccuracies caused by the tolerant policy technique. We present hereafter three steps to do the post-processing techniques.

1. **Recognition of type of correlation :** Suppose that after applying the *tolerant policy*, we identify that Φ and Ψ are correlated during $[a, b]$. Our

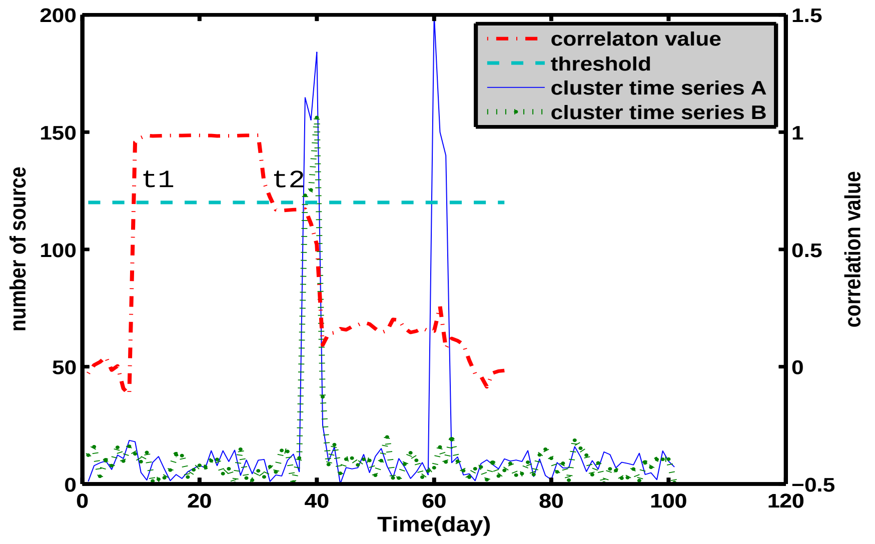


FIGURE 5.8 – The strict policy can not discover the correlation period in this case

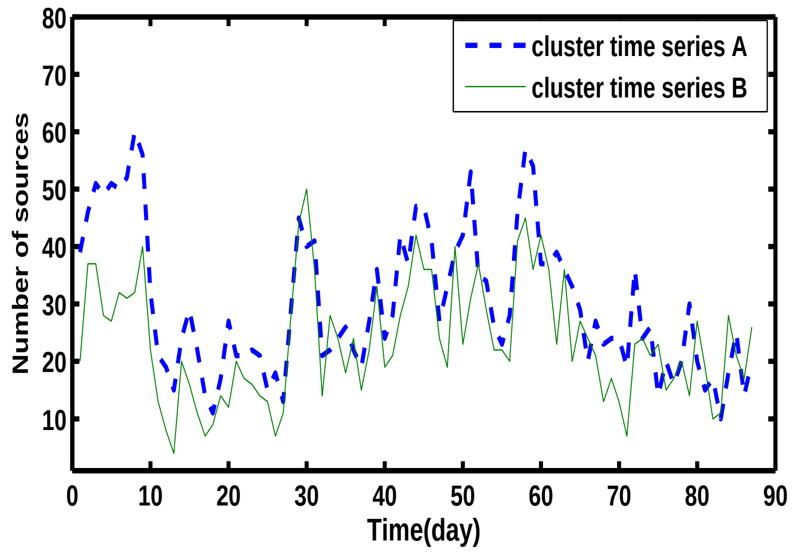


FIGURE 5.9 – Example of long correlation

observations show that the correlation of Φ and Ψ can be classified into one of the following two categories :

- *Short correlation :* The correlation of two time series within the time interval $[a, b]$ is mostly caused by some synchronized peaks of activities. Besides that, the two time series are not correlated at all. Figure 5.7 offers an example illustrating this case.
- *Long correlation :* Two time series are synchronized at almost every point in the time interval $[a, b]$. Figure 5.9 is an example illustrating this case.

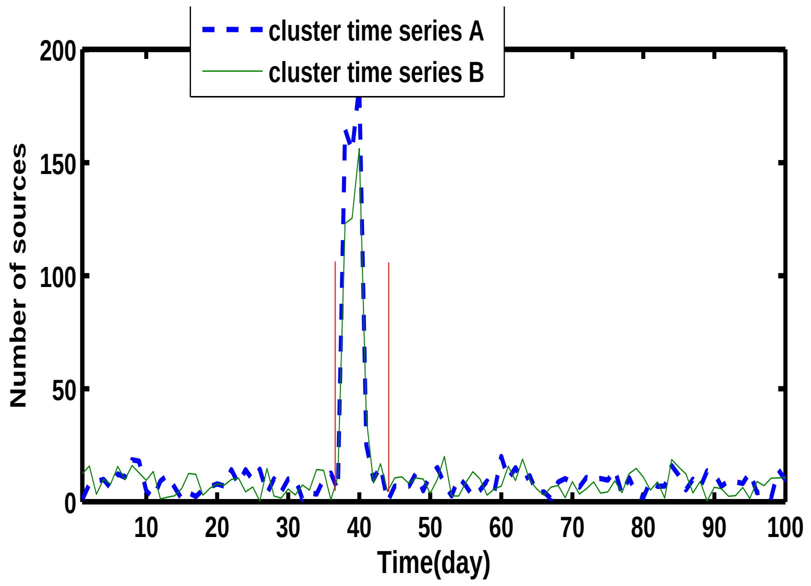


FIGURE 5.10 – Correlation detection of two time series 1 and 2. The strict policy says that the two time series are correlated from day 59 to day 63 whereas the tolerant policy says from day 29 to day 93

Since the inaccuracy occurs mostly in the case of short correlation, we apply the post-processing technique only to this family. We have tested the two following techniques to recognize the type of correlation of two time series in the period $[a, b]$.

– As mentioned earlier, in the case of short correlation, the correlation is caused mostly by the few synchronized peaks of activities. To recognize whether such peaks exist in a time series, we apply the technique as presented in Section 5.3. If there is a peak in the time series, we conclude that it belongs to the short correlation. Otherwise, we classify it into the long correlation family.

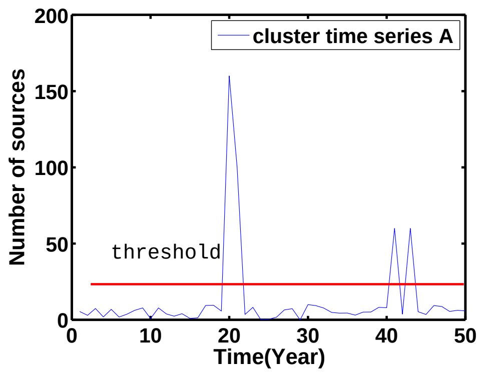


FIGURE 5.11 – Peak Peaking

– With the same arguments as before, we remove the most important data points from each time series. To recognize their type of correlation, we compute the correlation coefficient between the two residual time series. If the correlation computation yields a high value, we flag the correlation as long correlation. Otherwise, we class it into short correlation family.

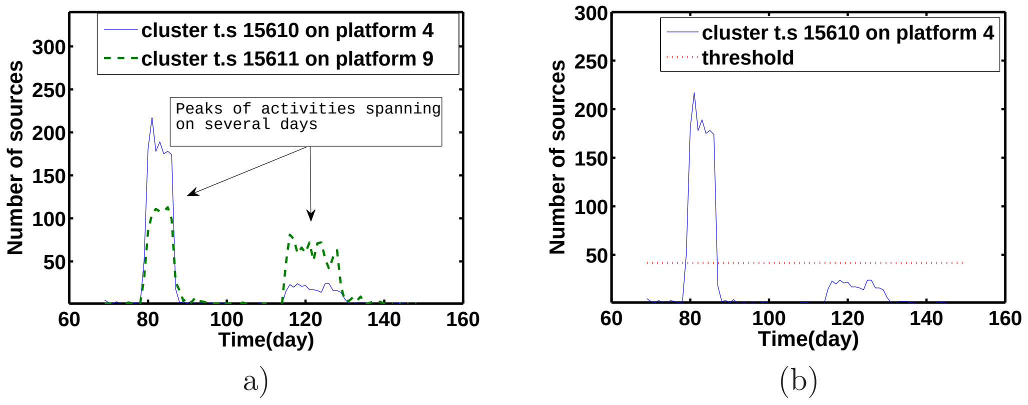


FIGURE 5.12 – (a) temporal correlated time interval b) peak picking technique misses the second peak

2. **Determination of potential correlated time interval :** Suppose that now we have two time series Φ and Ψ identified as being shortly correlated during $[a, b]$. As a remainder, the short correlation is due to the outliers existing in the time series. Our task now is to separate these peaks of activities on each time series during $[a, b]$. To identify the peaks, we can apply the peak picking technique presented in Section 5.3. We compare the data points of the attack trace with a threshold computed as a certain factor of the average of the population. Any data point greater than threshold is considered as a peak. Experiment shows that it works well especially when the peaks expand in a short time interval. For instance, Figure 5.11 shows an example of this technique. The threshold curve intersects with the time series curve at three places, the time series has three outliers. However, this approach does not give the expected results, neither when the important activities span on several days (not long enough though to make it belong to the *long correlation* class), nor when there are peaks of distinct heights, as in Figure 5.12a. In that specific example, there are two correlated peaks of activities of cluster 15611 and 15610 on platforms 4 and 9. There is almost no activity anywhere else during the time interval from day 60 to day 150 for these two clusters. Figure 5.12b represents the peak picking technique based on the average of the population. As we can observe, the method misses the second peak of activities since its values are less than the threshold. The reason for this is that the high peaks make the average value important.

   We have observed that, in the case of short correlation, the total spanning lengths, in term of number of days, of the important peaks of activities is much smaller than the temporal correlated time interval identified by the tolerant policy. To be sure that we can detect all the correlated time intervals, we choose a very conservative threshold of 40^{th} percentile[3] By doing so, the threshold does not depend on the outlier data points as far as their total spanning life is less than 60%. Since the 40^{th} is a conservative threshold, the amount of detected peaks of important activities will be greater than its real value. We show how to mitigate them in the next step. We use the peak picking algorithm represented in Algorithm1 and we give hereafter an example of the output of this function.

$$I_X = 0000\mathbf{111}00000\mathbf{11}00001\mathbf{11}0000$$

$$I_X(i) = \begin{cases} 0 & \text{if the } i^{th} \text{ element of X is below the threshold} \\ 1 & \text{if the } i^{th} \text{ element of X is above the threshold} \end{cases}$$

In this particular example, we have three peaks of activities detected. The first one starts from position 5 to position 7, the second one starts from position 12 to position 13, the third one starts from position 18 to position 20.

3. **Identification of correlated time intervals :** We identify all the periods where the elements from two time series are greater than the threshold. By doing so, we will reduce an important amount of wrongly detected peaks. For instance, let's consider the following outputs of the peak picking technique :

$$I_X = 0000\mathbf{111}00000\mathbf{11}00001\mathbf{11}0000$$
$$I_Y = 0000\underbrace{\mathbf{111}}_{T1}0000000000\underbrace{\mathbf{111}}_{T2}0000$$

As said earlier, there are three peaks (from position 5 to position 7, from 12 to 13, and from 18 to 20) in the time series X. And we have two peaks in the time series Y (from position 5 to position 7, and from position 18 to position 20). As a result, we have two correlated periods $P_1 = \{5, 7, X, Y\}$ and $P_2 = \{18, 20, X, Y\}$. Note that, the second peak (from position 12 to position 13) from the time series X is not considered as there is not corresponding period on time series Y.

Application of the above procedure to all the pairs of time series leads to the identification of a set of correlated pairs over different periods of time $P_i = (start, stop, \Phi, \Psi)$. Figure 5.13 illustrates the situation at the end of the first phase. In this illustrated case, the length T of the time series is equal to 9. A curve on the plot represents a correlated time interval of two time series, for instance, the plot shows that time series 1 and 2 (curve 1&2) are correlated from day 3 to day 6 or $P_1 = (3, 6, 1, 2)$, time series 7 and 8 (curve 7&8) are correlated from day 1 to day 8, $P_2 = (1, 8, 7, 8)$ etc.

3. the 40^{th} percentile of a time series X is the value at the position of 40% of the length of X on the sorted form of X

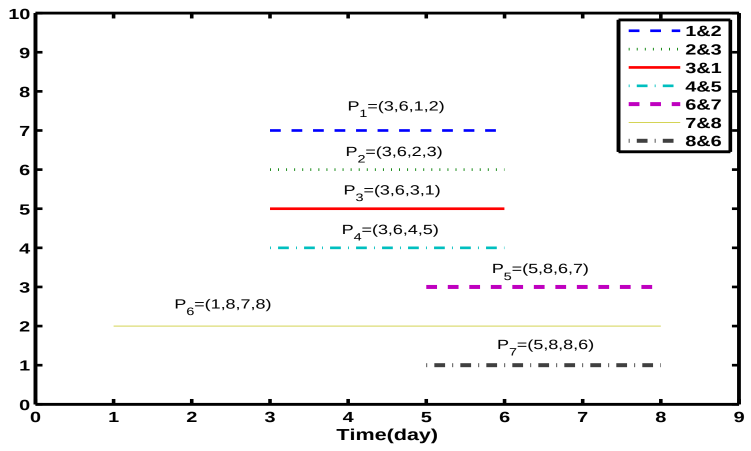


FIGURE 5.13 – Correlated pairs of time series over time, curve 1&2 says that time series 1 and time series 2 are correlated from day 3 to day 6...

5.4.2.3 Step 3 : Overlap Cutting

The two correlated pairs $P_1 = (start, stop, ts1, ts2)$, $P_2 = (start, stop, ts1, ts2)$ are said to be overlapping when the following conditions hold :

1. $P_{1,start} < P_{2,stop}$ and $P_{2,start} < P_{1,stop}$
2. $(P_{1,ts1} \cup P_{1,ts2}) \cap (P_{2,ts1} \cup P_{2,ts2}) \neq \emptyset$

There are two overlapping cases in Figure 5.13, i.e. between $P_5 = (5, 8, 6, 7)$ and $P_6 = (1, 8, 7, 8)$, $P_6 = (1, 8, 7, 8)$ and $P_7 = (5, 8, 6, 8)$. We are going to split the correlated pair $P_6 = (1, 8, 7, 8)$ into two correlated pairs $P_6' = (1, 5, 7, 8)$ and $P_6'' = (5, 8, 7, 8)$. The explanation for this is that there may be two activities. The first one involves only two time series 7 and 8 whereas the second one concerns three time series 6, 7, and 8. The whole procedure to split the overlapping of correlated pairs is described in Algorithm 2.

Alg. 2 : Overlap Cutting

```
Input :
        $P_1 = \{start, stop, X, Y\}$, $P_2 = \{start, stop, X, Z\}$ : two correlated periods
        with the overlapping of time series X
Output :
        $S$ : a set of correlated period
begin
  1     $S \leftarrow \emptyset$
  2     We create two new periods :
  3         $P'_1 = \{max(P_{1,start}, P_{2,start}), min(P_{1,stop}, P_{2,stop}), X, Y\}$
  4         $P'_2 = \{max(P_{1,start}, P_{2,start}), min(P_{1,stop}, P_{2,stop}), X, Z\}$
  5     and add them to $S$
  6     if $|P_{1,start} - P_{2,start}| > 0$
  7         $P'_{3,start} = min(P_{1,start}, P_{2,start})$
  8         $P'_{3,stop} = max(P_{1,start}, P_{2,start})$
  9         if $P_{1,start} < P_{2,start}$
 10             $P'_{3,ts1} = P_{1,X}$
 11             $P'_{3,ts2} = P_{1,Y}$
 12         else
 13             $P'_{3,ts1} = P_{2,X}$
 14             $P'_{3,ts2} = P_{2,Z}$
 15         end
 16         add $P'_3$ to S
 17     end
 18     if $|P_{1,stop} - P_{2,stop}| > 0$
 19         $P'_{4,start} = min(P_{1,stop}, P_{2,stop})$
 20         $P'_{4,stop} = max(P_{1,stop}, P_{2,stop})$
 21         if $|P_{1,stop} - P_{2,stop}| \geq 0$
 22             $P'_{4,ts1} = P_{1,X}$
 23             $P'_{3,ts2} = P_{1,Y}$
 24         else
 25             $P'_{4,ts1} = P_{2,X}$
 26             $P'_{4,ts2} = P_{2,Z}$
 27         end
 28         add $P'_4$ to S
 29     end
```

TABLE 5.1 – Common Correlated Time Intervals

$T_1 = [T_{start,1}, T_{stop,1}] = [5, 8]$	$S_1 = \{(6, 7), (7, 8), (8, 6)\}$
$T_2 = [T_{start,2}, T_{stop,2}] = [3, 6]$	$S_2 = \{(1, 2), (2, 3), (3, 1), (4, 5)\}$
$T_3 = [T_{start,3}, T_{stop,3}] = [1, 5]$	$S_3 = \{(7, 8)\}$

5.4.2.4 Step 4 : Common Correlated Time Interval Identification

Our next objective is to form the common correlated time interval $G = (\{T_1, S_1\}, \{T_2, S_2\}, \cdots \{T_k, S_k\})$. Each $G_i = \{T_i, S_i\}$ corresponds to a set P of correlated pairs, in which all pairs start at the same day $T_{i,start}$ and stop at the same day $T_{i,stop}$, and S_i contains all the time series in P, or $S_i = \cup P_{j,ts1}, P_{j,ts2} \forall P_j \in P$

Applying this to the case described in Figure 5.13, it leads to the identification of the three common correlated time intervals defined in Table 5.1.

5.4.2.5 Step 5 : Macro Attack Event Extraction

On the basis of common correlated time interval, we now need to group together all the time series that are mutually correlated with each other. To do that, we use a graph representation of the correlated pairs identified in the previous stage of our algorithm. Nodes in the graph represent the time series and if two time series are correlated in that period, their edges are connected. The problem corresponds to the identification of the connected subgraphs. We have tested the two following techniques :

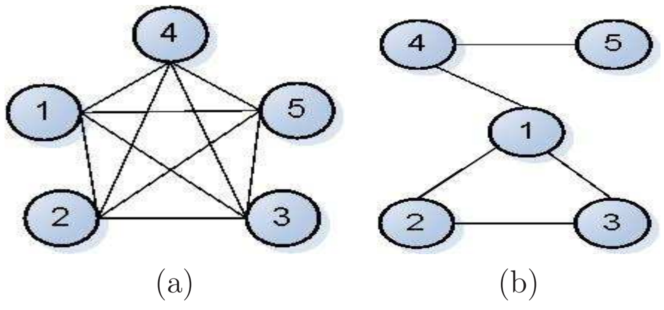


FIGURE 5.14 – group extraction technique : (a) clique, (b) connected component

- **Clique :** A clique in an undirected graph G is a set of vertices V such that for every two vertices in V, there exists an edge connecting the two, Figure 5.14a is an example of this. The clique extraction problem is an NP-complete one (the complexity is O($3^{n/3}$) with n is the number of nodes) [124].
- **Connected component :** A connected component in an undirected graph G is a set of vertices V such that there exists a path to connect any two nodes, Figure 5.14b is an example of this. This technique can resist to noise but in the same time, it may lead us to a situation where two non-correlated time series are grouped together. This is especially true for the nodes that are far away from each other on the graph.

TABLE 5.2 – Groups

$T_1 = [5,8]$	$S_1 = (6,7,8)$
$T_2 = [3,6]$	$S_2 = (1,2,3)$
$T_3 = [3,6]$	$S_3 = (4,5)$
$T_4 = [1,5]$	$S_4 = (7,8)$

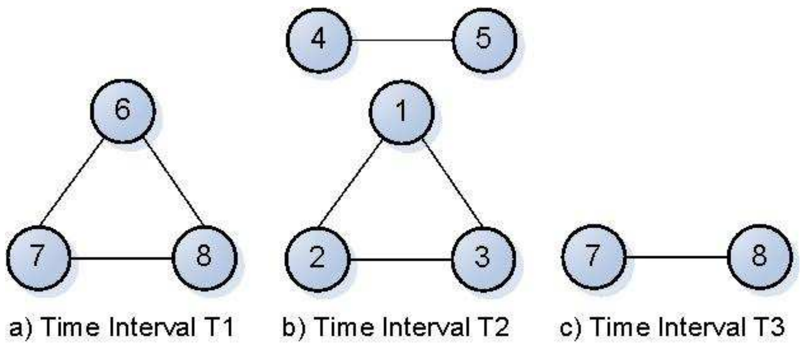


FIGURE 5.15 – Correlated groups extraction

Figure 5.15 depicts the graphs we obtain for the periods T_1, T_2, and T_3 extracted from Figure 5.13. According to this graph, we have four macro attack events $e_1 = (5, 8, \{6, 7, 8\})$, $e_2 = (3, 6, \{4, 5\})$, $e_3 = (3, 6, \{1, 2, 3\})$, and $e_4 = (1, 5, \{7, 8\})$

Complexity Analysis : This Section provides the complexity analysis to detect the macro attack events. We represent the cost in two cases : the theoretical one and the correlation sliding window one.

– *Theoretical Approach :* Given that we have N observed cluster time series of length T. Since we do not have any knowledge neither about which time series are involved, nor about when the macro attack events happen, we need to compare all the observed cluster time series to each other on all the sub time intervals $[a, b]$ of $[0, T]$. To discover macro attack events on any time interval $[a, b]$, we first need to compare all the observed cluster time series on this period to each other, i.e. to compute $\frac{N\times(N-1)}{2}$ correlation operations.
  After this step, we know how similar any two observed cluster time series are, thanks to this knowledge, we can build the similarity matrix, and apply the Step 5 as represented in Section 5.4.2 to identify the macro attack events existing in the time interval $[a, b]$. As showed in Section 5.4.2, this step has the computational cost of $O(3^{N/3})$. The total cost to detect all the macro attack events in any time interval $[a, b]$ is :

$$C_1 = O(N^2) + O(3^{N/3}) \tag{5.2}$$

  Since we have no knowledge beforehand when such macro attack events happen, in theory, we need to compute the correlation and identify the macro attack events for all possible sub intervals $(\frac{T\times(T-1)}{2})$ of $[0, T]$. As a result, the final computational cost is :

$$C_2 = (O(N^2) + O(3^{N/3})) \times O(T^2) \tag{5.3}$$

– *Correlation Sliding Window Technique :* Given a set of M time series with length of T days, a correlation sliding window of length L, for each time interval, we need $\frac{M\times(M-1)}{2}$ correlation operations. The number of windows we have is $T-L+1$. The cost for computing the correlation is :

$$C_3 = O(M^2T) \tag{5.4}$$

Suppose that we identify G common correlated time intervals, and K is the maximum number of observed cluster time series that are involved in the common correlated time intervals. The cost for extracting macro attack events is :

$$C_4 = G\times O(3^{K/3}) \tag{5.5}$$

The total cost of the *Correlation Sliding Window Technique* is :

$$C_5 = G\times O(3^{K/3}) + O(M^2T) \tag{5.6}$$

In practice, we have $K \ll M$ and $G \ll \frac{T(T-1)}{2}$, so $C_5 \ll C_2$.

5.4.3 Application of Correlation Sliding Window Technique

Although the pre-processing technique reduces an important amount of observed cluster time series, applying the correlation sliding window technique to detect macro attack events in the very large datasets may be costly. To deal with this, we propose hereafter two alternative solutions that can deal with large datasets with reasonable costs. The two approaches are called breakdown approach and aggregated time series approach. They are described as follows.

5.4.3.1 Breakdown Approach

This approach relies on the assumption that, in general, a macro attack event $e_i = (T_{i,start}, T_{i,end}, S_i)$ can only be classified into one of the following three classes :

1. Macro attack event e_i has only one cluster involved. In other words, S_i consists of a set of observed cluster time series of form $\Phi_{T',c,*}$. Such a macro attack event is referred as **distributed macro attack event**. We term *distributed* since such macro attack event concerns several locations.
2. Macro attack event e_i happens at only one observation viewpoint. In this case, S_i consists of a set of observed cluster time series of form $\Phi_{T',*,op}$. Such a macro attack event is referred as **localized macro attack event**. We term *localized* since such macro attack event concerns only one location but several clusters.
3. Macro attack event concerns more than one cluster and one observation viewpoint and can be split into several macro attack events belonging to the two earlier classes. Such a macro attack event is referred as **mixed macro attack event**.

With that in mind, given that M is the number of observed cluster time series $\Phi_{T,c,op}$, $C = (c_1, c_2, ...)$ is the set of distinct clusters, and $OP = (op_1, op_2, ...)$ is the set of distinct observation viewpoints, we proceed as follows to detect the three kinds of macro attack events as mentioned above :

– **Step 1, detection of distributed macro attack events :** we classify M observed cluster time series into $|C|$ sets $(D_1, D_2, ..., D_{|C|})$. The i^{th} set, D_i, contains only the observed cluster time series derived from cluster c_i, or $D_i = \Phi_{T',c_i,op}\ \forall op \in OP$. We then apply the correlation sliding window approach as presented earlier to each dataset D_i to detect the possible distributed macro attack events.

– **Step 2, detection of localized macro attack events :** we classify M observed cluster time series into, this time, $|OP|$ sets $(D'_1, D'_2, ..., D'_{|OP|})$, and the i^{th} set, D'_i, contains only the observed cluster time series derived from the observation viewpoint op_i, i.e., $D'_i = \{\Phi_{T',c,op_i}\}$ with $\forall c \in C$. We apply also the correlation sliding window technique to detect the eventual localized macro attack events existing in these datasets.

– **Step 3, detection of mixed macro attack events :** This step aims at re-grouping macro attack events identified earlier. Suppose that $E = (e_1, e_2, ..., e_n)$ is the set of all macro attack events detected during the two earlier steps. All macro attack events having the common lifetime (having the same T_{start} and T_{stop}) and having their time series correlated are merged. The newly obtained macro attack events are called mixed macro attack events.

Complexity Analysis : We will show later that the macro attack events often occur in a limited amount of locations and involve, most frequently, a limited number of clusters. That is the reason why we assume that the computational cost in Step 3 is negligible. We focus on Step 1 and Step 2. In Step 1, we apply $|C|$ times the correlation sliding window approach for $|C|$ sets of time series. Call $|D_i|$ the amount of time series of the i^{th} set, D_i, according to Equation 5.6, the computational cost of Step 1 is :

$$C_6 = \sum_{i=1}^{|C|} O(G_i \times 3^{K_i/3}) + O(|D_i|^2 T) \tag{5.7}$$

In which, G_i is the number of common correlated time interval of the dataset D_i, K_i is the maximum number of the observed cluster time series on all the common correlated time intervals. We expect that $G_i \ll \frac{T(T-1)}{2}$, and $K_i \ll |D_i|$.

Similarly, the computational cost of the Step 2 is :

$$C_7 = \sum_{i=1}^{|OP|} O(G'_i \times 3^{K'_i/3}) + O(|D'_i|^2 T) \tag{5.8}$$

The final computational cost of this approach is $C_8 = C_6 + C_7$. We expect that $C_8 \ll C_5$ since this approach does not compute the correlation between all the observed cluster time series. Besides that, we apply the *correlation sliding window* *technique* on the much smaller datasets (in Step 1 and Step 2) the cost will be less important.

5.4.3.2 Aggregated Time Series

Before going into the detail of the technique, we define precisely what is an aggregated time series.

Definition 9 ***An Aggregated Time Series*** *$\Phi_{T,op}$ is a function defined over a period of time T, T being defined as a time interval (in days). That function returns the amount of sources per day seen from the observation viewpoint op. The observation viewpoint can either be a specific platform or a specific country of origin. As a consequence, the aggregation time series $\Phi_{T,op}$ is the sum of all the observed cluster time series of form $\Phi_{T,*,op}$, or $\Phi_{T,op} = \sum^{\forall c \in clusters} \Phi_{T,c,op}$*

This approach aims at reducing this cost by adopting some assumptions about the characteristics of the macro attack events. In fact, we do not compare the observed cluster time series, but their aggregation. The question lies now in how to aggregate the attack traces. Our observation is that the attacks (or at least certain classes of attacks) are not uniformly distributed neither in their source nor in their destination [101, 69, 10, 22, 81], we make the assumption that if there are macro attack events (linked to certain attack tools) that target or come from certain places, these attacks will impact the total attack traces at those places. For instance, let's take the case of the destination of the attacks, our platforms observe a limited number of hits per day. If at some point in time two platforms become the target of the coordinated attacks, we make the assumption that this will significantly impact the overall corresponding *aggregated time series* of these two platforms during that period. Therefore, the method identifies groups of correlated aggregated time series over different time intervals. Obviously, if the intensity of the attack is not high enough to impact the aggregated time series of at least two locations, our method will miss it. When having the groups of correlated aggregated time series, we search for the root causes, i.e. the clusters that are responsible, if any, for the similarity of the aggregated time series in each group. Once we have found them, we verify that they do not also exist on other locations than the ones we initially had in the group under study. This can happen if the impact of these attacks on the other aggregated time series was not strong enough to include them in the group of correlated aggregated time series. By doing so, we hope to reduce the computational effort since the amount of aggregated time series is much smaller than that of all the observed cluster time series.

Hereafter is the detailed description of the method. It is important to notice that in the following text, we refer to the targets of the attacks, i.e. the platforms as the observation viewpoints, but all the steps can also be applied to the case where the observation viewpoints are, for instance, the countries of origin of the attacks.

1. ***Pairwise correlation of aggregated time series***
   To obtain the correlated groups of aggregated time series, we apply the techniques presented in Section 5.4.2. The only difference is that, in Section 5.4.2,

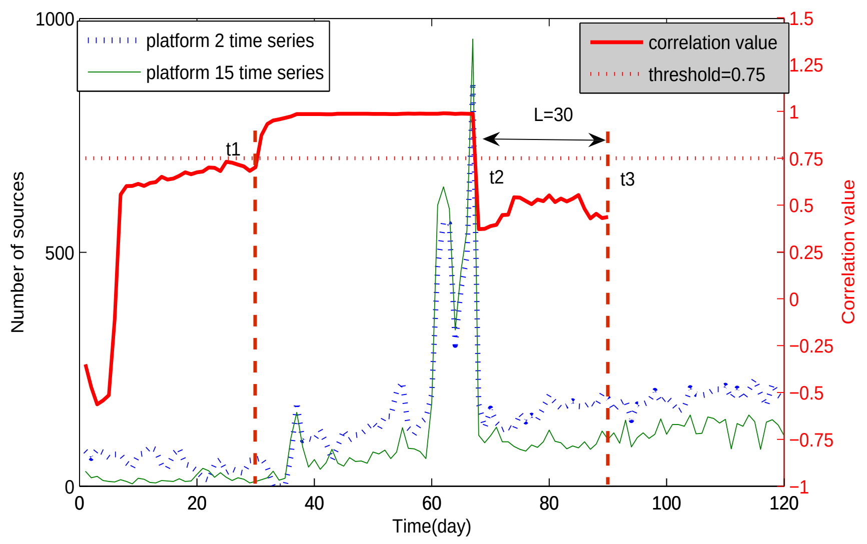


FIGURE 5.16 – aggregated time series 2 and 15 are correlated from day t_1 to day t_3

the correlation sliding window approach takes the observed cluster time series as input, in the present case, it takes the aggregated time series as input. To eliminate the impact introduced by the aggregated time series on the ability to identify the exact correlation period, we take advantage of the tolerant policy as described earlier in Section 5.4.2.

Figure 5.16 illustrates the first step of our procedure. The aggregated time series for platforms 2 and 15 are deemed correlated in the interval $[t_1, t_3]$ as their correlation vector is greater than the threshold of 0.7 in the period $[t_1, t_2] = [t_1, t_3 - L]$.

2. ***Root Cause Analysis***

The most intuitive explanation behind the existence of correlated groups of platforms is that those platforms are targeted by the same tool, launched from a diverse set of sources in a loosely coordinated way. In that case, the same cluster(s) should be found on each platform of the group as being the root cause of the correlation of the aggregated time series. We could, therefore, simply search for the root causes on one platform per group. However, as explained in [93, 85], multi-headed worms could hit platform X with cluster 1 and platform Y with cluster 2. Therefore, we take the stance of not assuming a priori that the traces left by a given attack tool are the same on the platforms that compose a correlated group. We thus look for the root causes behind the correlation independently for each platform in a correlated group. This means that for a period of T' days associated to a correlated group, we look, for each platform, for the set of observed cluster time series that are correlated with the aggregated time series. Here too, we use a sliding window as one can imagine that the aggregated time series are correlated due to two distinct, consecutive,

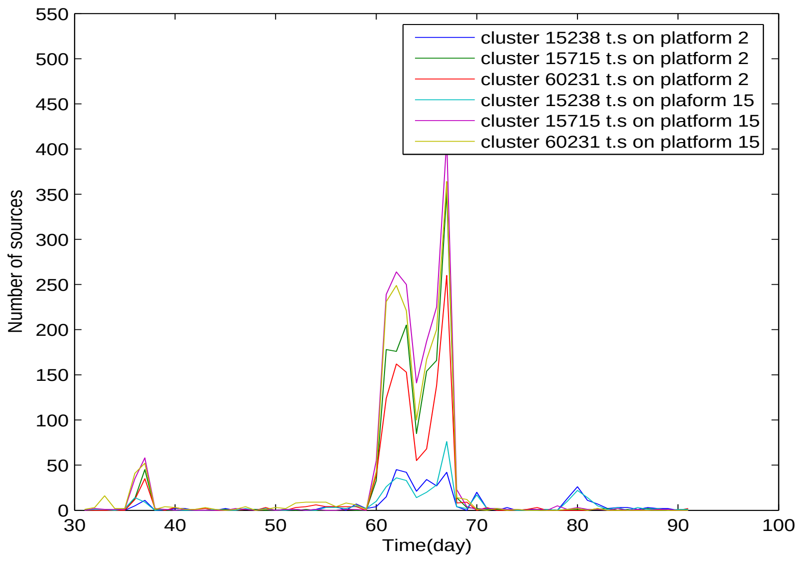


FIGURE 5.17 – cluster time series for the clusters uncovered during the root cause analysis for platforms 2 and 15

or overlapping phenomena. When obtaining a set of root cause time series, as a sanity check, we verify that they are all correlated to each other. If not, we will split them into several mutually correlated groups.

The correlated group in Figure 5.16 (between day 31 and day 91) provides an illustration of when the attack tool leaves the same fingerprint on each platform of a correlated group. Indeed, our root cause analysis technique identifies three clusters numbered 15238, 15715 and 60231 on both platform 2 and platform 15 as the root causes behind the correlation of the aggregated time series. Figure 5.17 depicts the observed cluster time series over the corresponding interval. Table 5.3 summarizes the correlation values obtained between the different observed cluster time series identified as root cause. As we can see, the correlation coefficients between those clusters are extremely high (greater than 0.85) in this period.

It is worth noting that we need to apply the set of techniques as described in Section 5.4.2 to the set of correlated root cause time series to detect exactly their correlated periods. As an example, in Figure 5.17, there are three different correlated periods.

3. ***Hidden Correlations***
The root cause analysis technique described above enables us to find a set of candidate clusters associated to each correlated group for each platform in that group. However, since we initially identify correlation based on the aggregated time series, it is possible that a tool targeted x platforms but the effect of the tool is only strongly influencing a subset of $y < x$ aggregated time series (e.g

TABLE 5.3 – Correlation coefficient between clusters

cluster t.s	2 15238	2 15715	2 60231	15 15238	15 15715	15 60231
15238-2	1.0000	0.8521	0.8422	0.8916	0.8631	0.8550
15715-2	0.8521	1.0000	0.9863	0.9248	0.9938	0.9908
60231-2	0.8422	0.9863	1.0000	0.9260	0.9873	0.9873
15238-15	0.8916	0.9248	0.9260	1.0000	0.9154	0.9121
15715-15	0.8631	0.9938	0.9873	0.9154	1.0000	0.9969
60231-15	0.8550	0.9908	0.9873	0.9121	0.9969	1.0000

due to the activity of other local malwares). To uncover all possible hidden correlations, we check if all clusters identified as root causes for a period of T' days for a correlated group are correlated with their siblings on the platforms that are not in the correlated group.

Complexity Analysis : Given S, the amount of the aggregated time series, the computational cost for correlation operations from Equation 5.6 now becomes

$$C_9 \;=\; O(S^2T) + O(G_1 \times (3^{K_1/3}) \tag{5.9}$$

This leads us to the identification of a certain amount P (with $P \ll T - L$) of periods in which we have a group of G_i (with $i = 1..P$ and for $\forall i|G_i| \ll S$) correlated aggregated time series. For each period, we have to find the clusters responsible for the identified similarity. In other words, for each period, we must compare the M observed cluster time series with, at maximum, S aggregation time series. If we define $G = max(G_i|i = 1..P)$ an upper bound of the cost of this operation can be given by $C_{10} = P \times G \times M$. Thus, the total cost of this method is equal to $C_{11} = C_9 + C_{10}$. In the general case, nothing ensures, *a priori*, that $C_{11} \ll C_8 \ll C_5$ but, as we expect that $G \ll S$ and $S \ll M$, this justifies the choice of this solution. Experimental results presented in Section 5.5.2 validate this choice.

5.5 Application and Validation

5.5.1 Dataset

5.5.1.1 Downtime Issue

To validate the methodology and the soundness of the results it generates, we need to apply it to a well defined dataset. For many reasons, platforms may be unable to capture data from time to time. When this happens, attack traces collected from those platforms exhibit peculiar increasing and decay shapes that are not related to the actual attack traffic. For instance, the top plot of Figure 5.18 shows the evolution of the number of sources observed on one of our platforms for the period from day 625 to day 660. For some reasons, this platform has been active only from day 639 to day 646. On the bottom plot, we show the observed cluster time series of this platform.

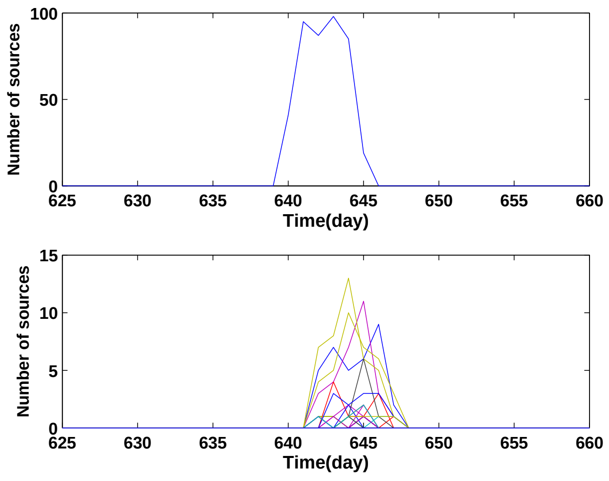


FIGURE 5.18 – top plot represents the amount of sources targeting the platform 49. For some reasons, this platform was down before the day 639 and some days after day 645. The bottom plot represents its compositional observed cluster time series. They are artificially correlated.

As we can see, they form some kinds of correlation that could eventually lead to artificial micro and macro attack events. To mitigate such artifacts, we must ensure that our platforms have experienced a limited amount of downtimes. Therefore, we build our dataset with data coming from platforms that have limited downtime over the whole experimental period.

5.5.1.2 Dataset Description

We have selected attack traces from 40 (out of 50) platforms for a period of 800 days. During this period, none of them has been down for more than 10 times and each of them has been up continuously for at least 100 days at least once. They all have been up for a minimum of 400 days over that period. Figure 5.19 offers a synthetic view of the availability of those platforms by plotting the cumulated distribution function of the cumulated uptimes of those platforms. The total amount of sources observed, day by day, on all these 40 platforms can be denoted by the initial time series TS over a period of 800 days. We can split that time series per country[4] of origin of the sources. This gives us 231 time series TS_X where the i^{th} point of such time series indicates the amount of sources, observed on all platforms, located in country X. According to our definition, these are the aggregated time series. And we represent by TS_L1 the set of all these aggregated time series. To reduce the computational cost, we keep only the countries from which we have seen at least 10 sources on at least one day. This enables us to focus on 85 (the set of corresponding countries is called $big_{countries}$), instead of 231, time series. We

4. The geographical location is given to us thanks to the Maxmind product, based on the IP address. However, some IPs can not be mapped to any real country and are attached to labels not corresponding to any country, e.g. EU,A1,..

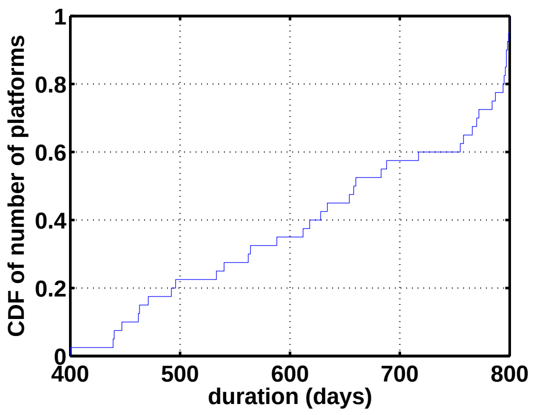


FIGURE 5.19 – CDF of the cumulated uptimes of 40 selected platforms

represent by TS_L1' this refined set of Level 1 time series. Then, we split each of these time series by cluster to produce the final set of observed cluster time series $\Phi_{[0-800),c_i,country_j} \forall c_i$ and $\forall country_j \in big_{countries}$. As a reminder, the i^{th} point of the time series $\Phi_{[0-800),X,Y}$ indicates the amount of sources originating from country Y that have been observed on day i attacking any of our platforms thanks to the attack defined by means of the cluster X. We represent by TS_L2 the set of all these observed cluster time series. In this case $|TS\_L2|$ is equal to 436,756 which corresponds to 3,284,551 sources.

As explained in Section 5.2, time series that barely vary in amplitude over the 800 days are meaningless to identify micro and macro attack events and we can get rid of them. Therefore, we only keep the time series that highlight important variations during the 800 days period. We represent by TS_L2' this refined set of Level 2 time series. In this case $|TS\_L2'|$ is equal to 2,420 which corresponds to 2,330,244 sources. Among them, we have 1,360 peaked time series, consisting of only 41,418 sources. As mentioned earlier, we tend to choose a conservative threshold in the pre-processing step to avoid putting the strongly varying and peaked time series in the class of stable time series, but we accept the false positive, i.e. to put the stable time series to the peaked time series class. And finally, we have 1,060 strongly varying time series, consisting of 2,288,826 sources.

We have done the very same splitting and filtering by looking at the traces on a per platform basis instead of on a per country of origin basis. The corresponding results are given in Table 5.4.

5.5.2 Analysis on the Detection Capacity

We want to analyze two factors that impact our capacity to detect micro and macro attack events. The first aspect concerns the choice of the method used, namely correlation sliding window, breakdown, or aggregated time series. We want to evaluate how those approaches perform when applied on the same dataset. This is the purpose of this section. The second factor to analyze is the impact of the obser-

TS consists of 3,477,976 sources		
OP	country	platform
$\|TS\_L1\|$	231	40
$\|TS\_L1'\|$	85 (94,4% TS)	40 (100% TS)
$\|TS\_L2\|$	436,756	395,712
$\|TS\_L2'\|$ sources	2,420 2,330,244 (67% of TS)	2,127 2,538,922 (73% of TS)
$\|TS\_L2\_P'\|$ sources	1,360 41,418	1,046 47,673
$\|TS\_L2\_S'\|$ sources	1,060 2,288,826	1,081 2,491,249

Table 5.4 – dataset description : TS : *all sources observed on the period under study*, OP : *observation viewpoint*, TS_L1 : *set of time series at country/platform level*, TS_L1' : *set of significant time series in* TS_L1, TS_L2 : *set of all cluster time series*, TS_L2' *set of refined time series of* TS_L2, TS_L2_P' *set of peaked time series*, TS_L2_S' *set of strongly varying time series*

vation viewpoints, namely what happens when we split the time series by country or by platform. We present this analysis in 5.5.3.

To compare our different approaches, we apply all of them to the set of 1,060 strongly varying time series derived from platform viewpoint. The results are presented in Table 5.5. It is not a surprise that the correlation sliding window approach has identified more macro attack events than the others. The aggregated time series based approach has detected the least amount of macro attack events. The reason would be that the impact of the attacks on the aggregated time series is not strong enough to make the corresponding aggregated time series correlated. To verify this explanation, we have counted the amount of sources for different approaches. The result confirms our assumption. In fact, although the aggregated time series approach detects only 41.5% of the macro attack events detected by the correlation sliding window approach, these macro attack events count for 78.4% number of sources detected by the correlation sliding window approach. The conclusion is that the aggregated time series approach can be used as long as we only are intended in detecting large phenomena. The breakdown approach is much better than the aggregated time series one. In fact, it detects 85.4% of the macro attack events detected by the correlation sliding window one, and in terms of number of sources, it finds 95% of the sources. For illustrative purpose, Figure 5.20 shows an example of macro attack event that the breakdown approach could not detect. In this particular example, there are three different clusters on three different platforms. We can not find any evidence that confirms the link between these attack traces. At this stage, we can not draw any conclusion about the validity of this macro attack event. More work needs to be done to clarify this. There are 92 macro attack events of this type.

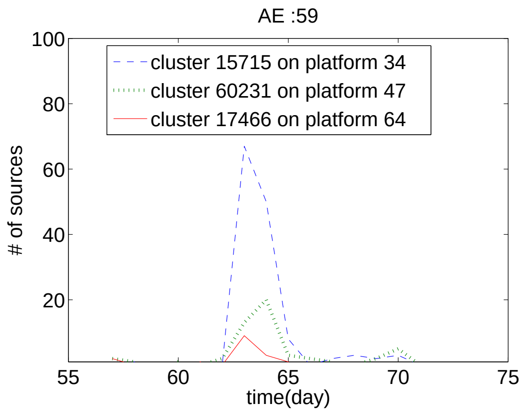


FIGURE 5.20 – Three clusters attack on three different platforms on three different ports (135/TCP, 5900/TCP, 445/TCP)

TABLE 5.5 – Overview of Results from several Techniques

	Correlation-Sliding Window	Breakdown	Aggregated-Time Series
# macro attack events	631	539	262
# of sources	566,661	538,194	444,389

5.5.3 Impact of Observation Viewpoint

5.5.3.1 Results on Attack Event Detection

To evaluate the impact of the observation viewpoint, we have applied the correlation sliding window technique and micro attack event detection technique presented in Section 5.3.1 to our 2 distinct datasets, namely $TS_{country}$ and $TS_{platform}$. The reason we apply the micro attack event detection is that the correlation sliding window can not detect phenomena that involve only one observed cluster time series. For the sake of efficiency, we apply the micro attack event detection only on the time interval of observed time series that do not belong to any macro attack events detected by the correlation sliding window technique. For the time series in $TS_{country}$, the detection of macro attack events by correlation sliding window M1 (resp. micro attack event detection M2) has found 549 macro (resp. 43 micro) attack events. The total amount of sources found in these attack events is 552,492 for the first method and 21,633 for the second one. Thus, all in all, sources participating to identified attack events account for 574,125 sources (corresponding to 16,5% of all sources contained in our initial dataset). Similarly, when working with the time series found in $TS_{platform}$, we end up with a total of 690 micro and macro attack events this time, containing 578,372 sources. The results are given in Table 5.6.

It is important to highlight that the micro and macro attack event concepts are introduced to facilitate the problem representation. They are, however, similar in the sense that all the attacking machines in a micro and macro attack event are in action with the same cause. Therefore, from now on, we use the term *attack event* to refer to both micro and macro attack event.

TABLE 5.6 – Result on Attack Event Detection

	AE-set-I($TS_{country}$)		AE-set-II($TS_{platform}$)	
	No.AEs	No.sources	No.AEs	No.sources
M1	549	552,492	631	566,661
M2	43	21,633	59	11,711
Total	592	574,125	690	578,372

No.AEs : amount of attack events
M1,M2 : methods presented in Sections 5.3,5.4

5.5.3.2 Analysis

The table highlights the fact that depending on how we decompose the initial set of traces of attacks (i.e the initial time series TS), namely by splitting it by countries of origin of the attackers or by platforms attacked, different attack events show up. To assess the overlap between attack events from two attack event sets *AE-set-I* and *AE-set-II* detected from different observation viewpoints we use the *common source ratio, namely csr*, measured as follows :

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

in which $e \in AE_{op}$ and $|e|$ is the amount of sources in attack event e, AE_{op} is *AE-set-I* and $AE_{op'}$ is *AE-set-II* (or vice versa).

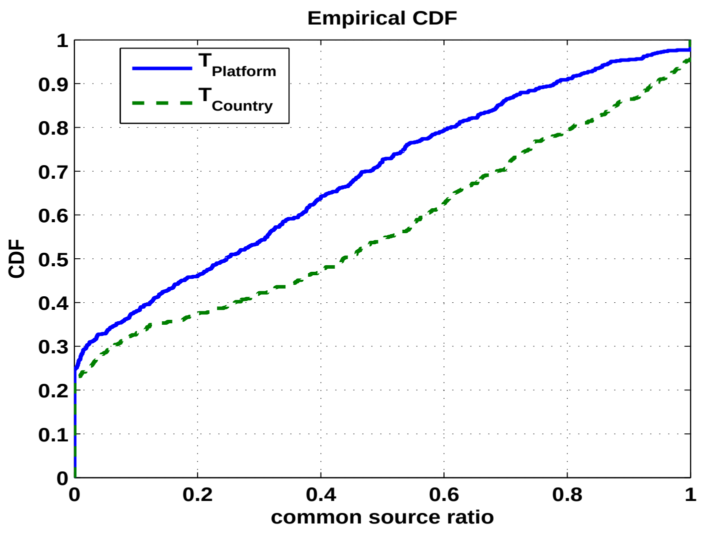


FIGURE 5.21 – CDF common source ratio

Figure 5.21 represents the two cumulative distribution functions corresponding to this measure. The point (x, y) on the curve means that there are $y*100\%$ of attack events obtained thanks to $T_{country}$ (resp $T_{platforms}$) that have less than $x * 100\%$ of sources in common with all attack events obtained thanks to $T_{platforms}$ (resp $T_{country}$).

The $T_{country}$ curve represents the cumulative distribution obtained in this first case and the $T_{platforms}$ one represents the CDF obtained when starting from the attacks events obtained with the intial $T_{platforms}$ set of time series. As we can notice, around 23% (resp. 25%) of attack events obtained with the $T_{country}$ (resp. $T_{platform}$) set of time series do not have any source in common with any attack event obtained when using the $T_{platform}$ (resp. $T_{country}$) set of time series. This corresponds to 136 (16,919 sources) and 171 (75,920 sources) attack events not being detected. In total, there are 288,825 (resp. 293,132) sources present in AE-Set-I (resp. AE-Set-II), but not in AE-Set-II (resp. AE-Set-I). As a final note, there are in total 867,248 sources involved in all the attack events detected from both datasets which correspond to 25% of the attacks observed in the period under study. It shows that the attacks on the Internet are not totally random anymore but more coordinated and strategical.

5.5.3.3 Explanation

There are good reasons that explain why we can not rely on a single viewpoint to detect all attacks events. They are described below.
Split by country : Suppose we have one botnet B made of machines that are located within the set of countries $\{X, Y, Z\}$. Suppose that, from time to time, these machines attack our platforms leaving traces that are assigned to cluster C. Suppose also that this cluster C is a very *popular* one, that is, many other machines from all over the world continuously leave traces on our platforms that are assigned to this cluster. As a result, the activities specifically linked to the botnet B are lost in the noise of all other machines leaving traces belonging to C. This is certainly true for the cluster time series (as defined earlier) related to C and this can also be true for the time series obtained by splitting it by platform, $\Phi_{[0-800),C,platform_i} \forall platform_i \in 1..40$. However, by splitting the time series corresponding to cluster C by countries of origins of the sources, then it is quite likely that the time series $\Phi_{[0-800),C,country_i} \forall country_i \in \{X, Y, Z\}$ will be highly correlated during the periods in which the botnet present in these countries will be active against our platforms. This will lead to the identification of one or several attack events.
Split by platform : Similarly, suppose we have a botnet B' made of machines located all over the world. Suppose that, from time to time, these machines attack a specific set of platforms $\{X, Y, Z\}$ leaving traces that are assigned to a cluster C. Suppose also that this cluster C is a very *popular* one, that is, many other machines from all over the world continuously leave traces on all our platforms that are assigned to this cluster. As a result, the activities specifically linked to the botnet B' are lost in the noise of all other machines leaving traces belonging to C. This is certainly true for the cluster time series (as defined earlier) related to C and this can also be true for the time series obtained by splitting it by countries, $\Phi_{[0-800),C,country_i} \forall country_i \in big_{countries}$. However, by splitting the time series corresponding to cluster C by platforms attacked, then it is quite likely that the time series $\Phi_{[0-800),C,platform_i} \forall platform_i \in \{X, Y, Z\}$ will be highly correlated during the periods in which the botnet influences the traces left on the sole platforms concerned by its attack. This will lead to the identification of one or several attack events.

The top plot of Figure 5.22 represents the attack event 79. In this case, we see that the traces due to the cluster 175309 are highly correlated when we group them by platform attacked. In fact, there are 9 platforms involved in this case, accounting for a total of 870 sources. If we group the same set of traces by country of origin of the sources, we end up with the bottom curves of Figure 5.22 where the specific attack event identified previously can barely be seen. This highlights the existence of a botnet made of machines located all over the world that target a specific subset of the Internet.

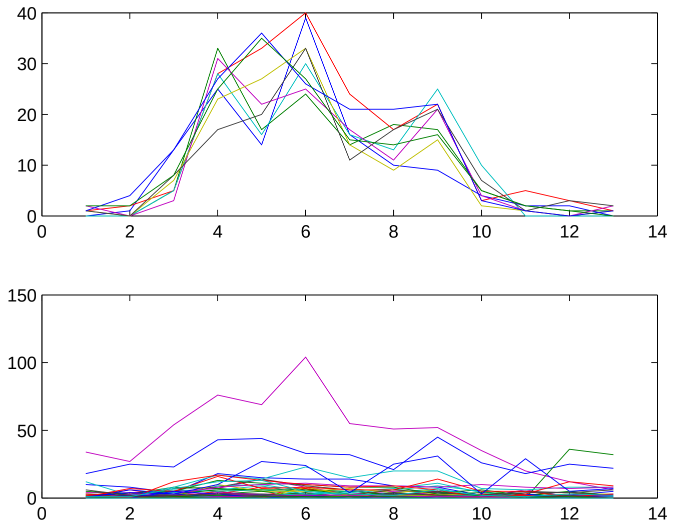


FIGURE 5.22 – top plot represents the attack event 79 related to cluster 17309 on 9 platforms. The bottom plot represents the evolution of this cluster by country. Noise of the attacks to other platforms decreases significantly the correlation of observed cluster time series when split by country

5.6 Summary

We have shown that it is possible to automatize the identification of the micro and macro attack events. To achieve this, we have adopted the following key assumption. The attacking sources acting under the same root cause have a particular distribution in terms of time and geographical location. This leads us to the observation that the attack events exist under form of peaks of activities or groups of correlated attack traces. Since detecting macro attack events requires a huge computational cost, we have discussed and implemented three alternative solutions for this purpose. The solutions are the balance between computational cost and ability to identify the attack events. All solutions were carefully designed to work with large datasets. We have validated our detection techniques on a real dataset collected from a distributed honeypot sensors. And the results conformed our expectations. Besides that, we have shown the impact of the clustering of the attack traces on our ability to detect micro and macro attack events. Depending on whether we used the origin or the destination of the attacks, we end up identifying different sets of attack events.

Chapter 6

CHARACTERIZATION OF ZOMBIE ARMIES

6.1 Introduction

We have shown in the previous Chapter how to detect the attack events existing in the attack traces collected by a distributed honeypot infrastructure. We have also discussed the impact of the observation viewpoints when detecting them. To justify all these efforts, in this Chapter, we provide diverse lessons that can be learned from the attack event concept. Concretely, in Section 6.2, we classify the attack events into three classes and then analyze them according to several characteristics. As we will show later, botnets are identified as the main cause of the attack events. We show how attack events help in better analyzing the botnets by bringing forward some important characteristics such as the lifetime of botnets, lifetime of infected machines in botnets, the kind of attack tools that infected machines possess. This is presented in Section 6.3.

6.2 Attack Event Characteristics

6.2.1 Attack Event Classification

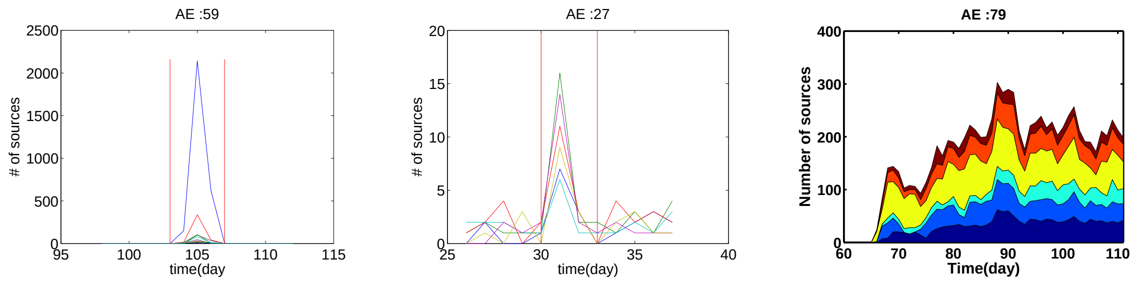
We believe that attack events are generated by different causes, for instance, one may be generated by a self propagating worm, the other may be traces of a static, remotely controlled botnet. As suggested in [129, 69], such coordinated attacks can be due to activities of worms, botnets and network misconfiguration. Sharing the same spirit, we classify the attack events that we have detected earlier into three classes : botnet, worm, and others. We name the third class *others* since we are not sure about the nature of this class of activities. We observe that there is a class of attack events made of attacks hitting only one platform, and on only one single and always the same IP address on that platform. A closer look at these attack events reveals that their attacking sources target only high port numbers, and most of them are used by clients of P2P networks. This somehow suggests that these attack events

Table 6.1 – Result on Classification of Attack Events

	AE-set-I	AE-set-II
Botnet	532	597
Worm	18	36
Others	42	57

are not caused by worms or botnets. In fact, different worms and botnets may have different strategies to choose their targets (hitlist, uniform, random mode...) but they always try to increase their spreading efficiency by contacting several machines. As mentioned in Chapter 3, our platform consists of three consecutive IP addresses. And given a large amount of sources in each attack event (mean sizes of attack events in *AE-set-I* and *AE-set-II* are 970 and 838 sources, respectively), *we expect that the attack events caused by botnets and worms hit all three IP addresses when they target a platform.* For this reason, we classify the attack events hitting only one IP address into the class *others*. It is important to notice that, authors in [69] also used the limited amount of IP addresses contacted to classify attacking sources in what they called "event probes" to the class *network misconfiguration*. While it is easy to differentiate attack events of class *others* from those of classes *botnet* and *worm*, it is not straightforward to differentiate activities caused by worms and those caused by botnets. In fact, we would expect that the source count of attacks left by a botnet has a sharp raise and a sharp decay as botnet receives order to probe. Whereas source count of attack traces left by a worm should have the exponential growing trend. However, if a botnet uses the pull mode to receive command, for example as it is the case of Phatbot, all the bots in that botnet do not start launching attacks at the same time. In fact, as mentioned in [131], Platbots : "[...] wake up every 1000 seconds to check for new commands. Given this behavior, rather than a sharp onset we instead might expect a steady rate of arrival over an interval of 10-20 minutes". In our case, we use the time step of one day, we hope this will eliminate the artifact of the pull mode. So, in our analysis, we consider all the attack events having the sharp growing and sharp decay as being generated by botnets. The remainders are attributed to activities caused by worms. The final result of the classification of attack events is represented in Table 6.1. As we can see, the majority of the attack events are caused by botnets. It is interesting to see that there are more attack events in class *others* than that in class *worm*, and this is true for the results obtained from both attack event sets *AE-set-I* and *AE-set-II*.

Figure 6.1a represents an example of attack event from class *others*. This particular attack event, consisting of 4054 sources from several observed cluster time series, targets port 50286/TCP on only one IP address of one platform located in China. Figure 6.1b shows the attack event 27, AE27, in which the cluster 15238 targets Netbios service (port 139/TCP) on IP addresses of five different platforms. With respect to the shape, the two attack events AE27 and AE59 look similar : sharp increase and sharp decay, and in both cases, the attacks last for a very short amount of time. However, AE27 hits several IP addresses on a well-known port and the AE59 hits only one IP address on a high port number. AE27 is therefore classified into the class *botnet*. Finally, Figure 6.1c represents the attack event 79, AE79, in which sources of the cluster 65710 attack the Windows Messenger Service port (1026/UDP) on six different platforms located in five /8 networks. The difference from both two previous examples is that in AE79 the attacks last for a long period of time.



(a)others (b)botnet (c)worm

FIGURE 6.1 – a) 4054 sources (belonging to several clusters) coming from Spain attack one IP address on high port number 50286/TCP b) Cluster numbered 15238 attacks against 5 platforms on Netbios Service (port 139/TCP) c) Cluster 65710 sends packets to 6 platforms against Windows Messenger Popup Service in a long period of time.

6.2.2 Characteristics of Attack Events

We want to verify the validity, the meaningfulness of the attack events identified, and, therefore see if other characteristics show their consistency. In the following, we use two sets of attack events *AE-set-I* and *AE-set-II* identified in Section 5.5.3.

Attacked services : We observe that the attack events belonging to both classes *worm* and *botnet* involve the similar list of services. In fact, in both cases, the attack events target common services such as Netbios (139/TCP, 445/TCP), RPC (135/TCP), Microsoft SQL Server (1443/TCP), VNC (5900/TCP), Symantec System Center Agent (2967/TCP), Windows Messenger Popup (1026-1028/UDP). For instance, Table 6.2 gives a detailed distribution of the services attacked by the 18 attack events belonging to the class *worm*. The first (resp. second) column represents the distribution of the services attacked by the attack events from *AE-set-I* (resp. *AE-set-II*). On the other hand, the attack events from the class *others* involve only high port numbers. Most of them used by eMule and eDonkey client, e.g. 4662/TCP. We actually do not have any good explanation for these phenomena. Tables 6.3 and 6.4 give a detailed distribution of ports attacked by the attack events of class *others* obtained from *AE-set-I* (resp. *AE-set-II*).

Lifetime of attack events : Lifetime of an attack event is defined as the time interval between the *start_at* and *end_at* of that attack event. As expected, attack events from both classes *botnet* and *others* have short lifetimes. In fact, almost all attack events from the class *others* last less than 3 days. Figure 6.2a shows the

Table 6.2 – Distribution of Attacked Services of Attack Events of Class Worm

AE-set-I	AE-set-II	service name
1	3	VNC (5900 TCP)
0	1	Scan, VNC (5900 TCP)
11	3	Symantec (2967TCP)
3	11	Microsoft Windows Messenger (1026/UDP-1027/UDP-1028/UDP)
1	6	Scanning (ICMP)
0	3	Scan, Netbios (139/TCP, 445/TCP)
1	0	Netbios (139/TCP, 445/TCP)
0	4	RPC (135)
1	4	MS SQL Server (1443)
0	1	MySQL (3306TCP)

Table 6.3 – Distribution of Ports Attacked by Attack Events (from *AE-set-I*) of Class *Others*.

#of attack events	Ports
28	eDonkey,eMule(4662/TCP 4672/TCP)
14	26912T 1755T 24653T 28238T 6342T 16661T 4857T 50286T 15264T 64264T 9763T 9661T 64783T 12293T

Table 6.4 – Distribution of Ports Attacked by Attack Events (from *AE-set-II*) of Class *Others*.

#of attack events	Ports
32	eDonkey,eMule(4662/TCP 4672/TCP)
25	18794T, 26912T, 24653T, 48080T, 21415T, 16661T, 19464T, 30491T, 4857T, (13208T,25801T), 50286T, 15264T, (33018T,64264T), 38266T, 5001T, 7690T, 6134T, 64783T, 64783T, 12293T, 12293T, 38009T, 64697T, 46030T, 10589T

CDF of lifetimes of attack events belonging to the class *botnet*. As we can notice, more than 90% of attack events from this class last less than 10 days. Whereas, attack events belonging to the class *worm* last for a long period of time. As shown in Figure 6.2b around 80% of attack events obtained from *AE-set-II* last for more than 50 days. It is also interesting to see that only 50% of attack events of class *worm* obtained from *AE-set-I* last more than 50 days. In other words, there is a difference in the lifetime of attack events belonging to the class *worm* detected by different observation viewpoints. More concretely, the lifetime of attack events detected by using the destination of the attacks is longer than that detected by using the origin of the attacking machines.

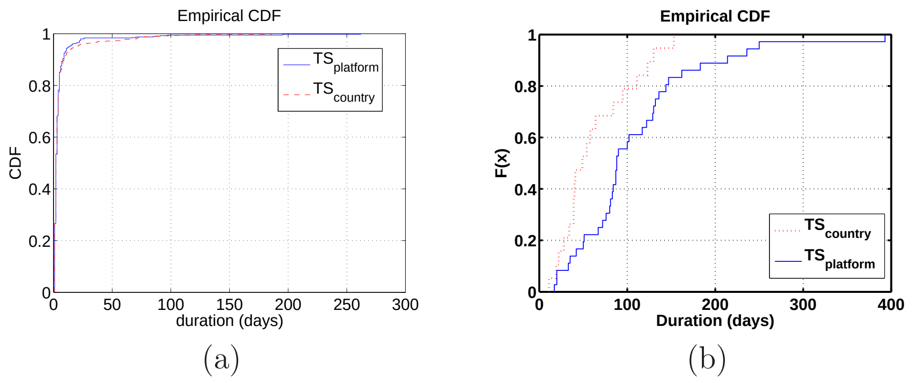


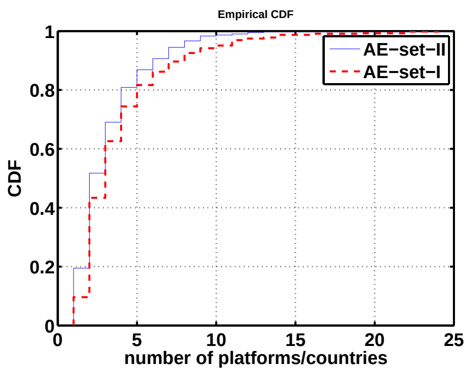
FIGURE 6.2 – a) CDF of lifetimes of attack events from class *botnet* b) CDF of lifetimes of attack events from class *worm*

Source and Target Distribution : Figure 6.3 shows the CDF of number of countries/platforms involved in the attack events for the classes *botnet* and *worm*. As we can observe, most attack events concern a limited number of observation viewpoints. In fact, in 80% of the cases, attack events involve less than 5 countries and platforms. In other words, attacking machines involved in attack events come from and attack a very limited number of locations in the IP space.

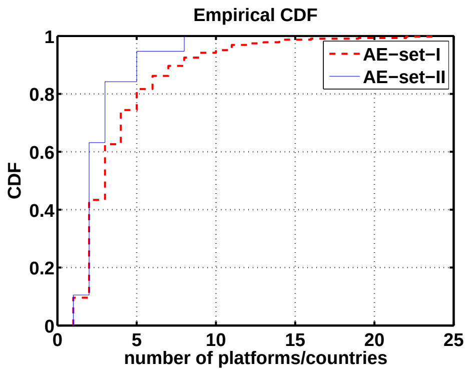
Finally, as said earlier, each attack event in class *others* always targets the same IP address. A closer look shows that 57 attack events in Table 6.4 attack only 5 distinct IP addresses from 3 platforms. As of now, we have no explanation for this strange phenomena.

Source behavior : As we have shown earlier, an attack event, consisting of several attacking sources, can involve more than one platform. It may be interesting to see whether these attacking sources do redundant tasks or if there is some assigned task for each source, or at least a mechanism that avoids the redundancy of work done by different attacking sources. To find the answers for this question, we look at the behavior of attacking sources through the following two aspects : the behavior of the attacking sources within one platform and the behavior of the attacking sources on several platforms. By behavior, we mean simply the number of honeypots and platforms contacted for the first, and second case, respectively.

To examine the behavior of the attacking sources within the platform, we proceed



(a)



(b)

FIGURE 6.3 – a) CDF of number of observation viewpoints from class worm. b) CDF of number of observation viewpoints from class botnet

as follows. Suppose that S is the set of all sources in an attack event, a source $s \in S$ contacts a set of platforms P_s, and N_i is the number of honeypots that the source s contacts on the platform $p_i \in P_s$. We compute $ms = \frac{1}{|P_s|} \sum^{p_i \in P_s} N_i$ as the mean of number of honeypots contacted by the source s per platform. The average number of honeypots (IP addresses) within a platform contacted by sources is $\frac{1}{|S|} \sum^{s \in S} ms$. Figure 6.4a shows the CDF of the average number of IP honeypots for all the attack events from the class *worm*. As we can see, in more than 80% the case, the sources contact in average more than 1 IP address. And in around 36% the cases, the sources contact all three IP addresses.

To examine the behavior of the attacking sources over several platforms, on the attack event basis, we compute, this time, the average number of platforms contacted by all the attacking sources from that specific attack event. Figure 6.4b shows the CDF of the average number of platforms contacted by sources for all the attack events from the class *worm*. As we can notice, in around 90% of the cases, attacking sources have contacted less than two platforms.

This result suggests that objective of the individual attacking source is always smaller than that of the attack event, and that there may be a mechanism that allows the attacking sources to avoid doing the redundant task within an attack event, but obviously, there are a lot of room to improve. This observation also holds with even stronger evidence in the case of botnets as represented in the two corresponding plots c and d of Figure 6.4.

As an example, attack event 79, as represented earlier in Figure 6.1b, consists of the attacks of cluster 65710 against Microsoft Messenger Service (port 1026/UDP) on six different platforms located in five /8 networks. In this particular attack event, a source contacts only one platform. In other words, we have 6 set of sources, in which sources in each set attack only one platform. Furthermore, all the sources hit all the three honeypots on the platform they contact.

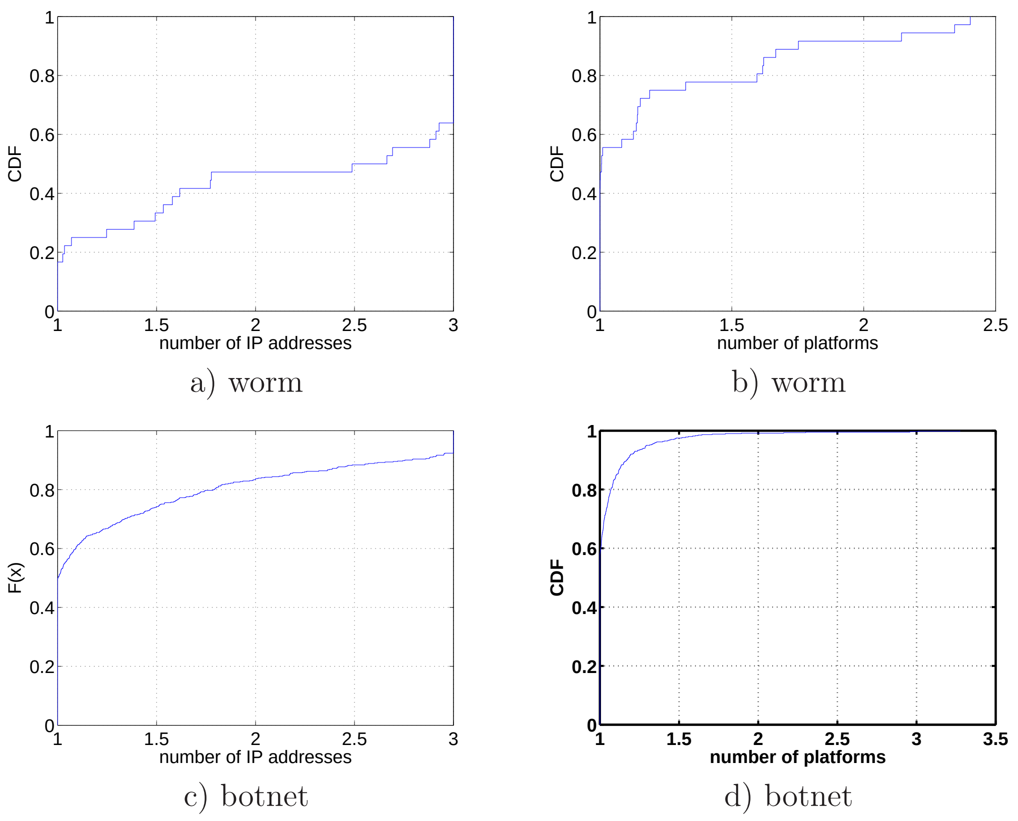


FIGURE 6.4 – Plot (a) (resp. c) represents the CDF of average number of IP within a platform contacted by the sources in case of class worm(resp. botnet). Plot (b) (resp. d) represents the CDF of average number of platforms within an attack event contacted by the sources in case of attack events belonged to class worm(resp. botnet)

6.3 Clustering of Attack Events

In the previous Section, we have analyzed the characteristics of attack events with respect to the class we had assigned them to, i.e. *worm*, *botnet*, and *others*. We have shown several characteristics of these classes of activities. In this Section, we want to go one step further on this process, i.e. to study the attack events that seem to be generated by the same root cause, e.g., a particular botnet. We show that by studying the attack events generated by such refined root cause, we can gain more insight on the attack processes that have generated these attack traces.

6.3.1 Method

6.3.1.1 Similarity Measures

We believe that there are attack events that are generated by the same (or related) root cause(s), e.g., the same botnet used to launch different attacks at different points in time. If we can find the prevalent characteristics of these attack events, we can group them together. Intuitively, if two attack events share an important amount of attacking machines, there is a large chance that they have the same root cause. With this in mind, we use the common IP addresses to measure the distance between the attack events. Of course, we do not expect this number to be very high. In fact, given a botnet, as time passes, it is reasonable to expect members of botnet to be cured while others join. So, if the same botnet attacks our honeypots twice over distinct periods of time, one simple way to link the two attack events together is by noticing that they have a large amount of IP addresses in common. More formally, we measure the likelihood of two attacks events e_1 and e_2 to be linked to the same root cause by means of their similarity defined as follows :

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

We will say that e_1 and e_2 are caused by the same (or related) root cause(s) if and only if $sim(e_1, e_2) > \delta$. This only makes sense for *reasonable* values of δ. In fact, the value of δ must ensure that $sim(e_1, e_2) > \delta$ is not due to the random factor. To build the foundation for choosing δ, we want to determine the probability that two given attack events e_1 and e_2 could share a certain number of IP addresses by chance. We denote the corresponding sets of distinct IP addresses of e_1 and e_2 as S_1 and S_2, in which e_1 and e_2 have n_1 and n_2 IP addresses, respectively. Suppose that A is the set of all visible IP addresses, and $N = |A|$. Visible IP addresses are the addresses that are globally accessible. The visible IP addresses are much smaller than 2^{32} due to firewalls and private networks [48]. We now try to compute the probability for which S_1 and S_2 have the overlap of n IP addresses. To do so, for a given S_1, we count the number of ways to build S_2 in general, and S_2 having n IP addresses in common with S_1. In the first case, in theory, there are $\mathrm{C}_{n_2}^{N} = \frac{N!}{n_2!(N-n_2)!}$ ways to choose S_2. For the second case, we divide A into two subsets : S_1 and $\bar{S_1}$, in which $\bar{S_1}$ is the complementary set of S_1 in the space A, or ($S_1 \cup \bar{S_1} = A$ and $S_1 \cap \bar{S_1} = \emptyset$). As a consequence, the cardinal of $\bar{S_1}$ is $N - n_1$. Since, for a set S_2 having n IP addresses in S_1, we have $n_2 - n$ IP addresses in $\bar{S_1}$, the number of ways to choose such S_2 is the product of the number of ways to choose $n_2 - n$ IP addresses from $\bar{S_1}$ and the number of ways to choose n IP addresses from S_1. More precisely, we have $\mathrm{C}_{n_2-n}^{N-n_1} = \frac{(N-n_1)!}{(n_2-n)!(N-n_1-n_2+n)!}$ ways to choose $n_2 - n$ IP addresses from $\bar{S_1}$. And we have $\mathrm{C}_{n}^{n_1} = \frac{n_1!}{n!(n_1-n)!}$ ways to choose n IP addresses from S_1. As a result we have $\mathrm{C}_{n}^{n_1} \times \mathrm{C}_{n_2-n}^{N-n_1}$ ways to choose such S_2. Finally, the probability that S_1 and S_2 share n IP addresses in common is :

$$\begin{aligned} \frac{\mathrm{C}_{n}^{n_1} \times \mathrm{C}_{n_2-n}^{N-n_1}}{\mathrm{C}_{n_1}^{N}} &= \frac{\frac{n!}{n_1!(n-n_1)!} \times \frac{(N-n_1)!}{(n_2-n)!(N-n_1-n_1+n)!}}{\frac{N!}{n_1! \times (N-n_1)!}} \\ &= \frac{n_1!n_2!(N-n_1)!(N-n_2)!}{n!(n_1-n)!(n_2-n)!(N-n_1-n_2+n)!N!} \end{aligned}$$

To give an idea on how this number looks like in reality, we can use it with some typical values. The mean size of attack events in *AE-Set-I* and *AE-Set-II* are 970 and 838 IP addresses, respectively. According to [48] the number of the visible IP addresses estimated is 108 millions IP addresses (corresponding to 3,6% of allocated IP addresses (2.8 billions)). According to the formula above, the probability that two attack events of size 1000 share one IP address by chance is 0.09% which is supposed to be very low.

6.3.1.2 Clustering of Attack Events

We now use the $sim()$ function to group together attack events into what we call action sets. Action sets are sets of attack events that are believed to be generated by the same root cause. To do so, we build a graph where the nodes are the attack events. There is an arc between two nodes e_1 and e_2 if and only if $sim(e_1, e_2) > \delta$. All nodes that are connected by at least one path end up in the same action set. In other words, we have as many action sets as we have disconnected graphs made of at least two nodes ; singleton sets are not counted as action sets.

We note that our approach is such that we can have an action set made of three attack events e_1, e_2 and e_3 where $sim(e_1, e_2) > \delta$ and $sim(e_2, e_3) > \delta$ but where $sim(e_1, e_3) < \delta$. This is consistent with our intuition that a particular threat (e.g. botnet, worm) can evolve over time in such a way that the machines associated to that threat can, eventually, be very different from the ones found the first time we have seen the same army in action.

6.3.2 Analysis of Attack Events From Class Botnet

We apply the method presented in Section 6.3.1 to cluster the attack events into action sets. In our dataset, as we have more attack events from the class *botnet*, we choose to focus mostly on this family in the rest of this Section. The basic behavior of bots is to receive a command from their *C&C* channel and execute it. Since we have no information regarding the *C&C* they obey to, we do not know if these machines are part of a single botnet or if they belong to several botnets that are coordinated. Therefore, to avoid any ambiguity, we write in the following that they are part of an *army of zombies*. An *army of zombies* can be a single botnet or a group of botnets the actions of which are coordinated during a given time interval.

To test the sensitivity of the threshold δ, we have computed the amount of action sets for the two sets of attack events for different values of δ. The result is represented in Figure 6.5. It is important to notice that when we increase the value of δ, two processes are taking place. On one hand, for a small value of δ certain nodes remain connected together but, as δ increases, the initial graph loses arcs and more disconnected graphs appear, i.e., more action sets show up. In the other hand, some action sets also disappear with a growing δ value. This is due to the fact that some graphs are broken into isolated nodes that are not counted as attack sets anymore. If the creation speed is greater than the extinction speed the curve will go up, otherwise, it will go down.

As we can see, at first, for the value of δ from 1% to 6%, the amount of action sets increases rapidly. After this point since the extinction speed is greater than the creation speed, the curve mostly goes down. One would suggest to use $\delta = 6\%$ as an ideal threshold, but it may not always be the good choice. Since at that value of δ, the speed of creating action sets stays always high, it means that there may exist the weak links within the action sets. As a consequence, we may merge two or more unrelated action sets together. To be conservative, we choose the threshold δ a little bit greater than this value ($\delta = 8\%$) in hoping that we can avoid merging unrelated action sets together.

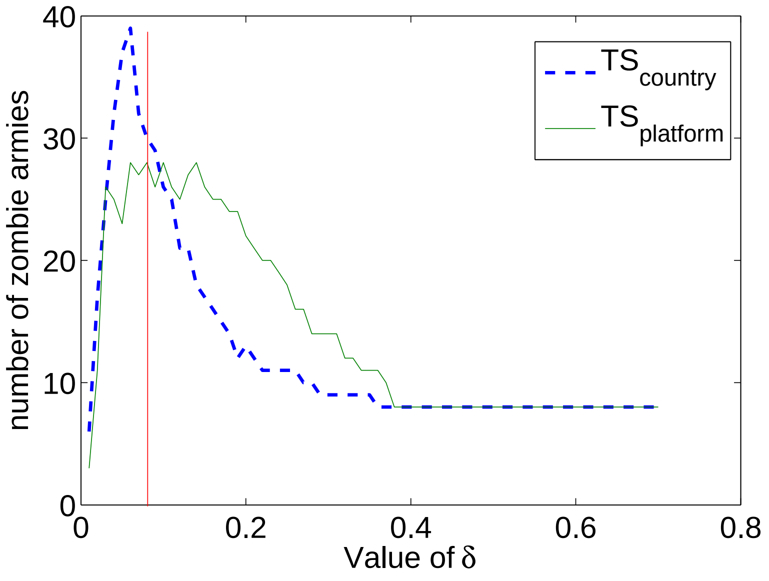


FIGURE 6.5 – sensitivity check of threshold δ

With that value of δ, we have identified 30 (resp. 28) zombie armies from AE-set-I (resp. AE-set-II). They contain a total of 181 (resp. 234) attack events. Figure 6.6 represents the distribution of size (in number of attack events) of the zombie armies. Its top (resp. bottom) plot represents such distribution of armies obtained from AE-set-I (resp. AE-set-II). We can see that the largest amount of attack events for an army is 50 in both cases, whereas 20 (resp. 18) armies have been observed only two times.

6.3.2.1 Lifetime of Zombie Armies

Figure 6.7 represents the cumulative distribution of lifetimes of the zombie armies obtained from *AE-set-I* and *AE-set-II*. According to the plot, around 20% of zombie armies have lasted for more than 200 days. In the extreme case, two armies seem to have survived for 700 days! Such result seems to indicate that either i) it takes a long time to cure compromised machines or that ii) armies are able to stay active for long periods of time, despite the fact that some of their members disappear, by continuously compromising new ones.

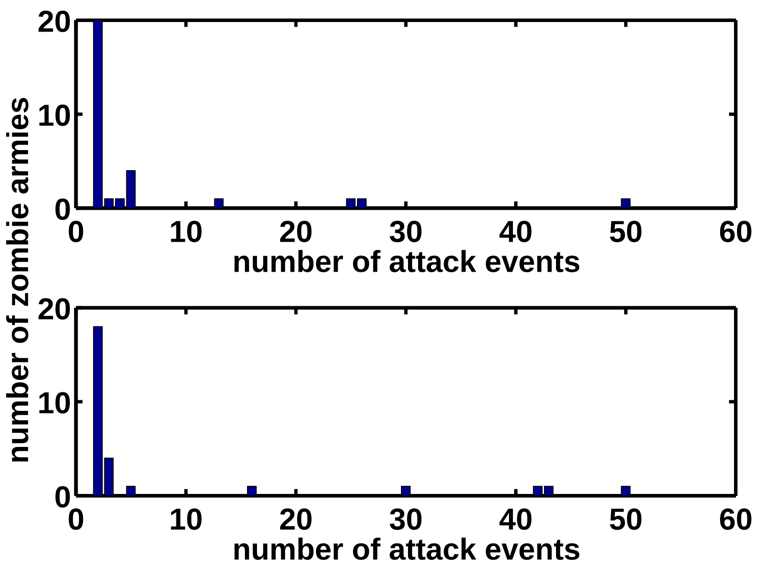


FIGURE 6.6 – Distribution of Zombie Army Size

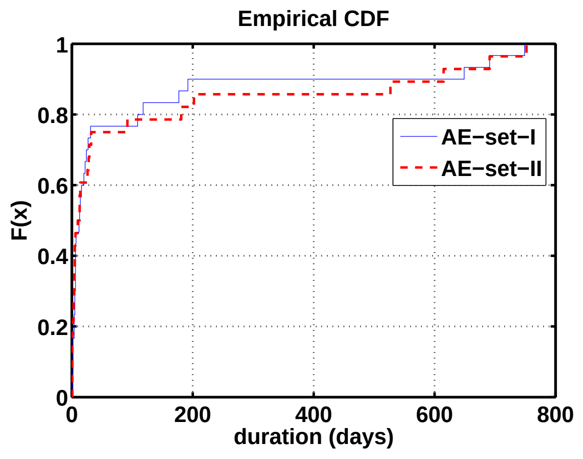
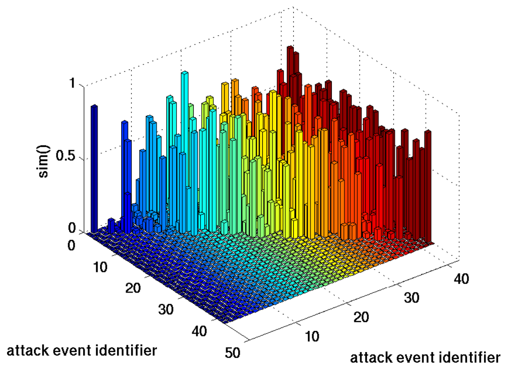


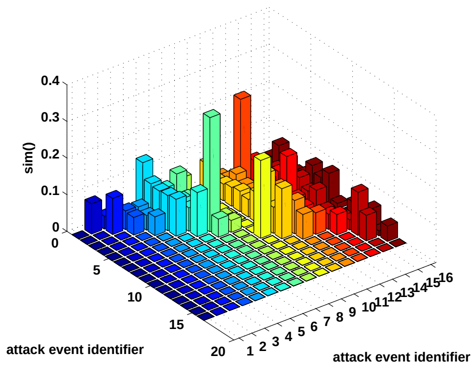
FIGURE 6.7 – CDF duration

6.3.2.2 Lifetime of Infected Host in Zombie Armies

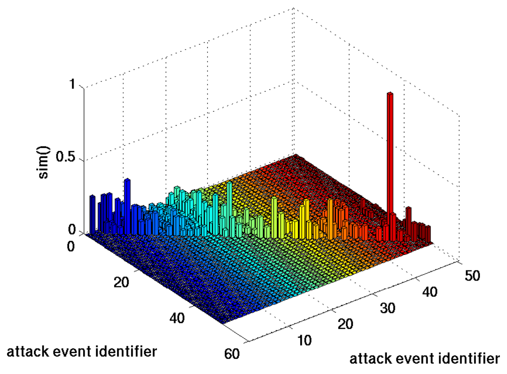
In fact, we can classify the armies into two classes as mentioned in the previous Section. For instance, Figure 6.8a represents the similarity matrix of zombie army 27, ZA27. To build this matrix, we first order its 42 attack events according to their appearance. Then we represent their similarity relation under a 42×42 similarity matrix $\mathscr{M}$. The cell (i,j) represents the value of $sim()$ of the ordered attack events i^{th} and j^{th}. Since $\mathscr{M}$ is a symmetric matrix, for the sake of visibility, we represent only half of it. As we can see, we have a very high similarity value between almost all the attack events, around 60%. This is also true between the very first and the very last attack events. It is important to notice the time interval between the first and the last activities observed from this army is 754 days! Similarly, Figure 6.8b represents the similarity matrix of zombie army 24, ZA24. This zombie army has also a very long lifetime (625 days), and all the attack events share an important amount of IP addresses with each other. However, in ZA24, the similarity values are much smaller than those in ZA27.



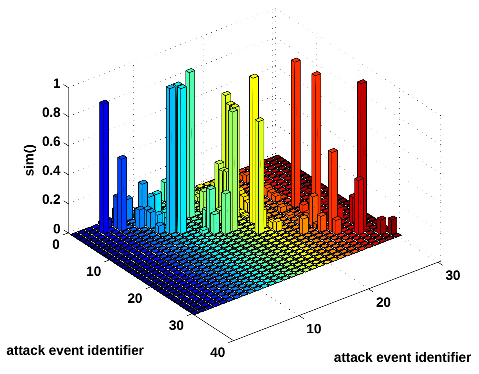
a) Zombie Army 27



b) Zombie Army 24



c) Zombie Army 28



d) Zombie Army 11

FIGURE 6.8 – Renewal rate of zombie armies

Figure 6.8c represents the opposite case, the similarity matrix of the zombie army 28 (ZA28), consisting of 50 attack events. As we can notice, the important values are near the main diagonal of $\mathscr{M}$[1]. It means that the attack event i^{th} has the same subset of infected machines with only few attack events happening not far from it in terms of time. And the lifetime of this army is 536 days! Finally, Figure 6.8d represents a similar example (zombie army 11, ZA11) as ZA28, but as we can see, the values near the diagonal of the similarity matrix are much higher. It is clear, from these four cases, that the composition of armies evolves over time in different ways. This can be seen thanks to the attack events and zombie armies identification process developed in this thesis. More work remains to be done in order to understand the reasons behind these various strategies.

1. The diagonal itself, of course, is not computed since sim(i,i)=1 $\forall i$.

6.3.2.3 Attack Tools Analysis

We want now to study the attack tools used by the infected hosts. Our fundamental assumption is that all behaviors of the infected machines are pre-programmed. Therefore, we can derive the nature of attack tools by observing the behavior of the attacking machines. We classify them into the following five classes : Single attack vector attack tool, variant signature attack tools, fingerprint attack tools, multi-attack vector attack tools, multi-headed attack tools. They are described hereafter.

1. **Single attack vector**
   Some IP addresses observed, always show the same behavior every time. For instance, attack event 102, represented in Figure 6.9, consists of 142 attacking machines that have launched 471 attacks against the four platforms 21, 27, 32, and 49. All these 471 attacks belong to the cluster 150691. According to our definition, an attacking machine in this case, can be observed, at maximum, four times. It means that a majority of these 142 attacking machines have launched the same attack, against more than one platform. Besides that, 47 out of these 142 attacking machines have been observed in another attack event (in a different time period), namely AE108, and there they have launched 89 attacks, all belonging to cluster 150691 by our clustering algorithm, targeting 4 platforms.

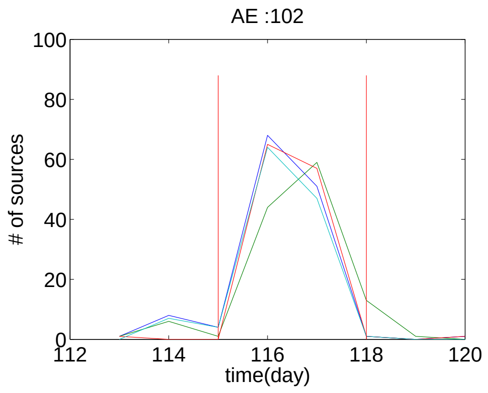


FIGURE 6.9 – Cluster 150691 targets four platforms 21, 27, 32, and 49

In total, we have observed the single attack vector behavior from attacking machines from 17 (out of 28) armies obtained from *AE-set-II*. The detail is presented in Table 6.5.

Army-Id	Single-Attack Vector	FingerPrint	Variant-Signature	Multi Attack Vectors	Multi-Headed
1	X				
2	X				
3	X				
4	X				
5	X				
6			X		
7	X				
8	X				
9	X				
10				X	
11		X	X		
12					X
13	X				
14		X			
15	X				
16	X				
17	X				
18	X				
19		X			
20	X				
21		X			
22		X			
23	X				
24		X		X	
25	X				
26			X		
27			X	X	
28	X		X	X	

Table 6.5 – Analysis of attack tool of armies obtained from AE-set-II

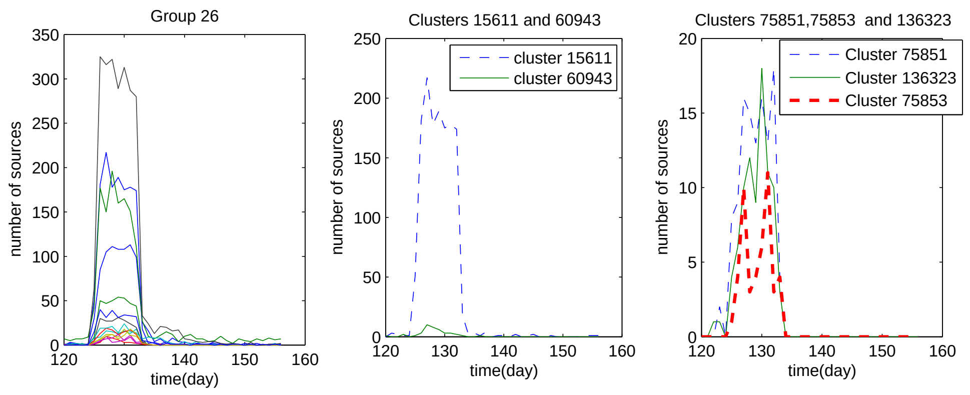


Figure 6.10 – Example variant worm

2. **Variant signature attack tools**
   Our clustering algorithm classifies sources into clusters on a basis of a set of attributes such as the number of packets sent by the sources to our platforms, the ports sequences, the number of virtual hosts contacted,etc. Not all attack tools have a deterministic behavior. Some may probe ports in a random order, a variable number of times, etc. As a result, traces left by such tools will appear in distinct clusters that will appear in correlated groups. In this specific dataset, we found three reasons for which clusters can be "split".

   The first one is that they have contacted a different number of targets. One cluster contacts only 1 honeypot and the other cluster contacts two honeypots. By our observation, two-honeypot-contacted clusters have a smaller number of sources than the one-honeypot-contacted clusters. It can be explained as follows : if one source randomly chooses its target in a network, the probability for it to hit only one of our machines is much higher than to hit two (or even three) of them. As an example, the left plot of Figure 6.10 represents the attacks of all cluster time series related to AE26. The middle plot of Figure 6.10 represents only the attacks of two clusters 15611 and 60943 on platform 5. Cluster 15611 contacts 1 honeypot and cluster 60943 contacts two honeypots.

   The other case is that the attack tool sends different amount of packets each time it attacks our platform. The right plot of Figure 6.10 represents the attacks of three clusters 75851, 75853 and 136323 also on platform 5. The three clusters have the same ports sequence :

   $ICMP|445T|139T|445T|139T|445T$. The difference resides in the number of packets sent by the sources in each of these clusters.

   In total, we have observed the variant signature behavior from attacking machines from 5 (out of 28) armies obtained from *AE-set-II*. The detail is presented in Table 6.5.

3. **Fingerprint attack tools**
   OS fingerprint is a well-known attack tactic. The idea is that before launching the attack, the attacker checks the type of target system it faces and

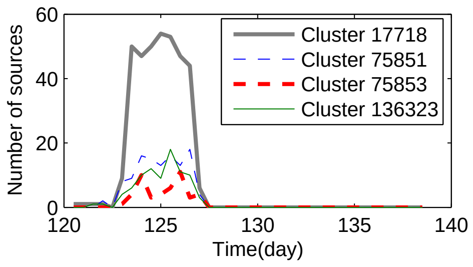


FIGURE 6.11 – Example Fingerprint worm

then launches, or not, the appropriate attack. We have found attack tools that automatized this idea. We call them "fingerprint attack tool". If a fingerprint attack tool learns that it is attacking a non vulnerable host (w.r.t its attack model), it gives up. On our platforms, we have deployed two kinds of virtual machines : Windows and Linux, the fingerprint attack tools will leave different traces on these two platforms if they adapt their behavior as a function of response received to previous packets. In terms of ports sequences, fingerprint attack tools may leave two different ports sequences on two kinds of virtual machines. One ports sequence may be the prefix of the other. For instance, we plot 4 clusters 75851, 75853, 136323, and 17718 of platform 5 from, again, attack event 26 on Figure 6.11. The three clusters numbered 75851,75853 and 136323 (resp. 17718) have the corresponding ports sequence $ICMP|445T|139T|445T|139T|445T$ (resp. $ICMP|445T$). Cluster 17718 is mostly observed on the Linux machine (296 sources). There are only 64 sources that sent packets to the other two windows machines. The three other clusters however, are only observed on the two windows machines (251 sources in total). The explanation is that since port 445TCP is closed on the Linux machine, the attack tool is "intelligent enough" not to try port 139 TCP since it knows that the target is not vulnerable w.r.t its attacks. The fact that 64 sources have contacted the two Windows machines but have given up can probably be explained by packet losses, either in the network (e.g, packet losses, firewall filters,etc..) or at the host (e.g, congestion while launching too many scans in parallel).

In total, we have observed the fingerprint behavior from attacking machines from 6 (out of 28) armies obtained from *AE-set-II*. The detail is presented in Table 6.5.

4. **Multi-attack vector attack tools**
These sophisticated attack tools can break into target machines using several different techniques. This, by itself, is not new. The Morris worm [118], in 1988, already had this feature. It was propagating using attacks against three different services : rshd, fingerd and sendmail. The Morris worm, after having selected a target, was trying all three attacks, one after another, interrupting the process only in the case of a successful intrusion. Several other worms have, since then, used the same strategy. They all are fairly easy to identify thanks

to the known sets (or sequences) of attacks they try against their targets. That leads to idea of having dynamic behavior for the attack tools. This idea has been proposed by Nazario et al in [79] in 2001. And by monitoring the botnet activities, authors of [40] have observed that it is quite frequent that "[...] the bots are instructed to download a piece of software from the Internet and then execute it". We term this class of attack tools as *multi-attack vector attack tools*. We have observed this at several occasions. For instance, attack event 195 targets only port 9661/TCP and attack event 299 targets only port 6211/TCP, and they share 77 IP addresses in common. It means that these 77 attacking machines are capable of attack with multiple attack vectors.

In total, we have observed the multi-attack vector behavior from attacking machines from 4 (out of 28) armies obtained from *AE-set-II*. The detail is presented in Table 6.5.

5. **Multi-headed attack tools**
Multi-headed attack tool is a special case of multi-attack vector attack tool. In fact, as defined in [93], a multi-headed attack tool has many attack vectors and uses them dynamically. The services targeted are usually different. The multi-headed attack tools allow the malwares to have a large chance to propagate. Actually, the malwares can still propagate even if some of the vulnerabilities they exploit are patched. Furthermore, since multi-headed attack tools exhibit varying activities it makes it harder for IDS to detect them. As an example, Figure 6.12a represents attack events 89. We have observed "three groups" of attacking machines that attack three different services. i.e., Netbios Service (139/TCP), Microsoft SQL Server (1443/TCP), and VNC (5900/TCP). Figure 6.12b represents the attack event 221. There too, we observe the same set of services attacked, and Microsoft SQL Server and VNC are also more attacked than Netbios Service. Attack event 89 and attack event 221 share 55 IP addresses in common, in which 28 IP addresses are observed keeping their behavior. In fact, 12 machines always attacked port 1443/TCP and the other 16 always attacked port 5900/TCP, and 27 machines have changed their behavior. More precisely, 16 machines have attacked port 1443/TCP in attack event 89, turned to attack port 5900/TCP in attack event 221, and 11 machines do the opposite. We actually do not see the machines attacking Netbios Service in the first place (on attack event 89) coming back in the attack event 221. This may be explained by some bias in the probability of choosing this attack vector.

We have observed the multi-headed behavior from attacking machines from only one (out of 28) armies obtained from *AE-set-II*. The details are presented in Table 6.5.

6.3.2.4 Attack Capacity of Zombie Armies

Earlier, we have studied the attack tools launched by the infected machines individually. In this Section, we focus on a broader perspective, i.e., the attack capacity of the zombie armies. By attack capacity, we refer to the amount of different

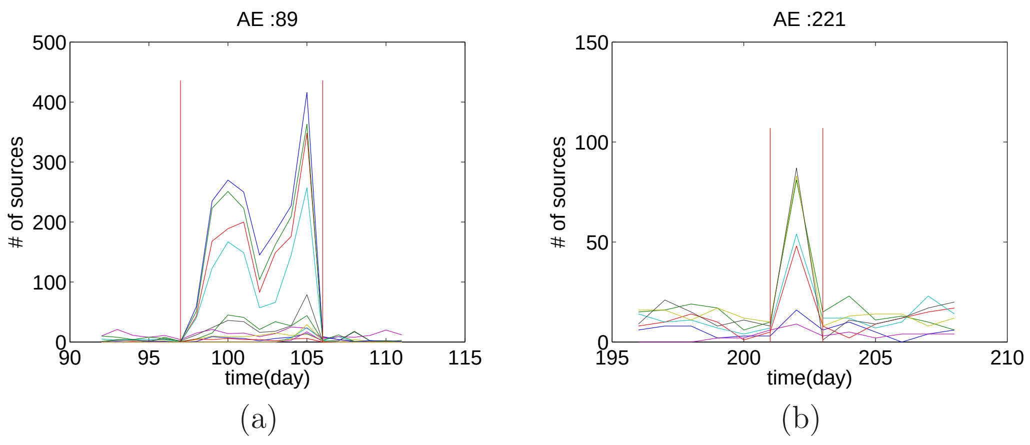


FIGURE 6.12 – a) attack event 89, b) attack event 221

attacks that a given army is observed launching over time. The top (resp. bottom) plot of Figure 6.13 represents the distribution of the amount of distinct clusters per zombie army detected in attack event set *AE-set-I* (resp. *AE-set-II*). In both cases, we observe that most of zombie armies exhibit multiple attack traces (corresponding to multiple clusters). In the extreme case, one zombie army has more than 100 distinct clusters. The large amount of distinct clusters can be due to the side-effect of the multi-signature of the attack tools, and the fingerprint attack tools. But it is more likely due to the update behavior of botnets. In fact, as observed in [69], "[...] The botmasters appear to ask most of the bots in a botnet to focus on one vulnerability, while choosing a small sub-set of the bots to test another vulnerability". Figure 6.14 represents the evolution of the number of distinct clusters over time of ZA27 and ZA28. In both cases, the point (x,y) on the curve means that up to the x^{th} attack event, we observe in total y distinct clusters. As we can see, in both cases, the bots keep trying new cluster during their whole lifetime. More specifically, when observing closely curve "zombie army 27" in Figure 6.14 we see that ZA27 tries a new cluster in each attack event, almost. The corresponding list of port sequences it has tried is

5900/TCP, 135/TCP, 2967/TCP, 139/TCP, ICMP, 80/TCP, 445/TCP, 1433/TCP, 4899/TCP, 5901/TCP,18886T, 1026/UDP, 445T139T445T139T445T. This is a clear case of multi-attack vector tool that we see evolving over time.

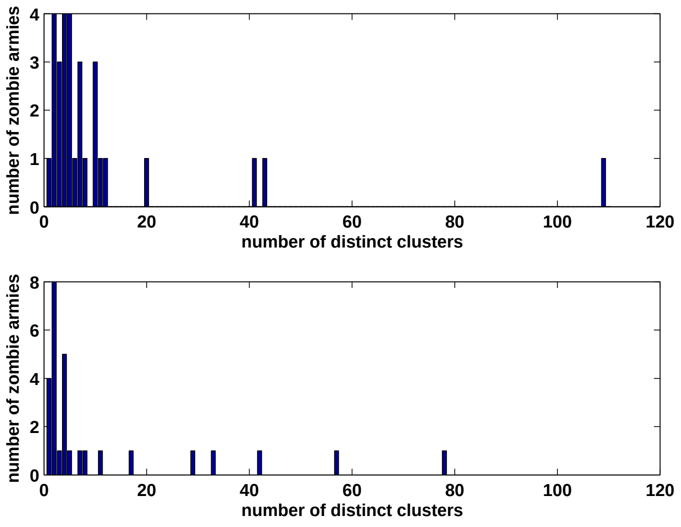


FIGURE 6.13 – Attack capacity of zombie armies

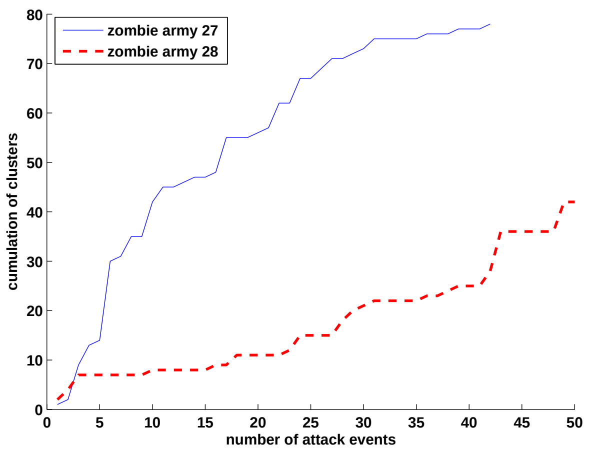
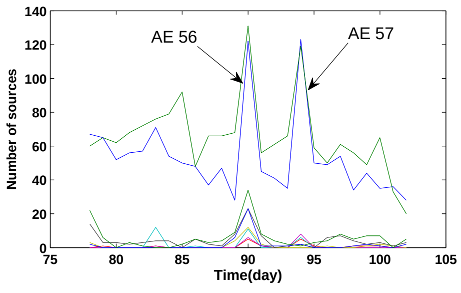


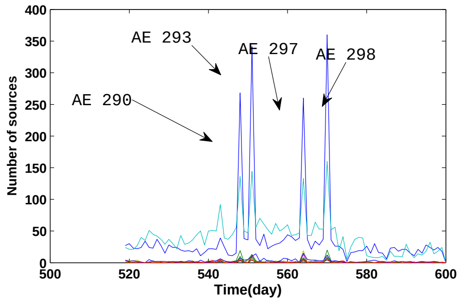
FIGURE 6.14 – Renewal rate of infected host of zombie armies

6.3.2.5 Illustrated Examples

Example 1 : Zombie army 24, ZA24, is an interesting example which has been observed attacking a single platform. However, 16 distinct attack events are linked to that army ! Figure 6.15a presents its two first activities corresponding to the two attack events 56 and 57. Figure 6.15b represents four other attack events. In each attack event, the army tries a number of distinct clusters such as 13882, 14635, 14647, 56608, 144028, 144044, 149357, 164877, 166477. These clusters try many combinations of Windows ports (135 TCP, 139 TCP, 445 TCP) and Web server (80 TCP). The time interval between the first activities and the last activities is 625 days !



(a)



(b)

FIGURE 6.15 – 6 attack events from ZA24

Example 2 : The zombie army 27, ZA-27, is another large army we have detected. In fact, it consists of 42 attack events (already mentioned in Section 6.3.2.2). As an example to show its activities, the top left plot of Figure 6.16 represents the attack event 12 left by this army. The target is the platform 26, and in this example, 85 attacking machines send the ICMP scan on three honeypots. The top right plot shows the attack event 307 that sends ICMP packets to the two platforms 26 and 41. In this particular attack event, every source hits only 1 single IP address. As a side note, the two attack events share 48 IP addresses in common. The bottom left plot of Figure 6.16 shows the attack events 454, this attack event hits 4 platforms (26, 13, 50, and 57) and the last one hits 26, 45, 53, 56, 57. To show how close these attack events are, Table 6.6 shows the number of common IP addresses between

TABLE 6.6 – Common IP addresses of four attack events 12, 307, 454, and 483

	12	307	454	483
12	85	48	60	36
307	48	93	63	37
454	60	63	370	47
483	36	37	47	108

these four attack events. For instance, attack event 483 shares 37 IP address with AE 307, and 454 and 483 share 47 IP addresses... Last but not least, the interval between the first and the last attack event issued by this zombie army is 753 days.

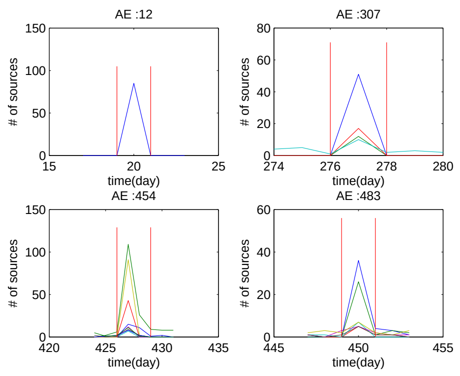
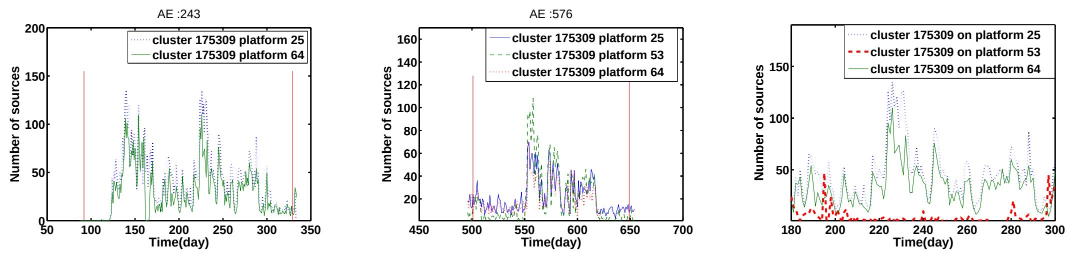


FIGURE 6.16 – 4 attack events from zombie army 27

6.3.3 Analysis of Attack Events from Class Worm

Clustering of attack events : Applying the clustering techniques as described earlier in Section 6.3.1, we obtain three action sets, in which two attack sets have 3 attack events and the last one has 2 attack events. As an example, one of the groups consists of three attack events 229, 243 and 328 that target ports 1433/TCP (MS SQL Server), 2967/TCP (Symantec service) and 5900/TCP (VNC service). The three attack events share the same lists of targets, i.e., platforms 25 and 64. As a reminder, since these attack events belong to an action set, they must share an important amount of sources in common. In fact, attack events 229 and 243 have 462 sources in common. Attack events 243 and 328 share 209 sources in common. This suggests that there is something in common in the attack processes that have generated these attack events.

Activity Discrimination : an attack tool may be exploited by different groups of attackers. We show that by correlating the attack traces, we can differentiate the activities associated to a particular attack tool. As an example, Figure 6.17a represents attack event 243 that shows the activities of cluster 175309 against Symantec Agent (port 2967/TCP). In this particular attack event, we observe that the activities involve only two platforms namely 25 and 64, and they both belong to the same class A. Whereas, Figure 6.17b represents the attack event 576, AE567 that targets the same two platforms (25 and 64) and also platform 53. Figure 6.17c represents the activities of cluster 175309 against three platforms 25, 53, and 64 on a sub period of attack event 243. As we can see, during this period, there is almost no activity of cluster 175309 on the platform 53. This is why cluster 175309 against platform 53 was not included within AE243, as opposed to AE567. We have checked that this phenomena was not a side-effect of the fact that the corresponding platform was down. The possible reason could be that the traffic has been filtered before reaching our platforms [22]. This could also be an indication of the fact that the two attack events have different root causes.



(a) (b) (c)

FIGURE 6.17 – Plot (a), Plot (b), Plot (c)

6.3.4 Analysis of Attack Events from Class Others

Similarly to the previous case, we apply the technique as described in Section 6.3.1 to two attack event sets *AE-set-I* and *AE-set-II*. When starting from the platform viewpoint, we obtain two action sets made of 2 and 8 attack events respectively. Whereas in the case of the country viewpoint, we obtain one action set of 7 attack events. This suggests the existence of stable processes that are the root causes of these attack events. However, it is impossible for us to infer the root cause of these phenomena. It may be due to the misconfiguration of P2P networks. Another potential cause could be the application of concepts described in [78, 35], in which authors show how to abuse P2P networks to generate DDoS attacks against an arbitrary victim. Due to the lack of concrete evidence, we can not draw any conclusion about these phenomena.

6.4 Conclusion

In this Chapter, we have shown, by looking at the attack events that our method can generate semantically meaningful results. We have validated this experimentally by using the two sets of attack events detected earlier in Chapter 5. More precisely, with respect to the cause of attack events, we have shown that we can classify them into three classes of activities : namely *worm*, *botnet*, and *others*. On the basis of class of activities, we have studied several characteristics of its attack events such as the lifetime of attack events, behavior of attacking sources, attacked services, distribution of platforms and countries involved in attack events. For instance, attacking and attacked machines involved in attack events are very location specific, i.e. each attack event concerns a few countries and platforms. Also, lifetime of attack events belonging to class *botnet* is very short in comparison to that belonging to class *worm*. Besides that, in taking advantage of data mining techniques, we have proposed a method to group together related attack events into what we called action sets. We have shown that studying the action sets indeed brings deeper understanding about the threats. For instance, in the case of botnets, we have discovered that botnets may exist for a very long period of time, over 700 days ! It is interesting to see that botnets adopt several strategies for renewing their zombie machines. Also, some botnets managed to survive for a very long period of time even though the renewal rate of their zombie machines is extremely high. We have also shown that we can deduce the kind of attack tools employed by the zombie machines in botnets. Indeed, we have shown that there are several kinds of advanced and sophisticated behaviors infected machines can have such as fingerprint, multi-attack vector, multi-headed. We have also observed that botnets frequently change their attack vectors in different attack campaigns. It is important to highlight that our solutions are easy to deploy and do not rely on any assumption about the communication protocol used by botnets. Our hope is to provide insights into the bigger picture of today's (and yesterday's) botnets activities. This kind of knowledge will help us to have a more precise perception on the current threats our computers are facing, and thus, to build a better countermeasure.

Chapter 7

CONCLUSION AND PERSPECTIVES

7.1 Conclusion

This thesis started by stating that it is important to understand the modus operandi of the attack processes from the attack traces, and that this task is difficult. The challenges we face when addressing this problem are due to the lack of knowledge about how and when the attacks are launched. It is therefore difficult to know whether two attacking sources are related (or launched by the same attack process) or not. While we need to study the attacking sources in their context in order to figure out how the attack process behind them operates. Ideally, we can classify the attacking machines into groups with respect to their root cause, i.e. the attack process triggering the attacks. In this thesis, we termed the groups of related attacking sources as micro and macro attack events. To identify and analyze such micro and macro attack events, we proposed to use classical signal processing and data mining techniques.

With respect to the thesis statement that opened this work, we have shown that it is possible to automatize the identification of the micro and macro attack events. To achieve this, we have adopted the following key assumption : The attacking sources acting under the same root cause have a particular distribution in terms of time and geographical location. This leads us to the observation that the attack events exist under form of peaks of activities or groups of correlated attack traces. As shown in Chapter 5, detecting macro attack events requires a huge computational cost, we have discussed and implemented three alternative solutions for this purpose. The solutions express the tradeoff between computational cost and ability to identify the attack events. All solutions were carefully designed to work with large datasets. Since the detection of macro attack events is based on the time series correlation technique, we have studied several similarity measures to identify the most appropriate technique for our application context. We have validated our detection techniques on a real dataset collected from a distributed honeypot sensors. And the results conformed our expectations.

As a validation of our approach in understanding the attack traces, we have proved that the attack traces can be classified into the following three families based on their activity level. The first family consists of attack tools that are almost constantly used in time. The second family consists of attack tools that are used from time to time over a period of couple of days. The last family consists of attack tools that are rarely used more than once and always during one or two days only. Besides that, studying the attack events detected by our techniques, we have shown that we can identify the cause and several characteristics of the attack events. In fact, with respect to the cause, attack events can be classified into the following three categories : *worm*, *botnet*, and *others*. We have also analyzed several characteristics of attack events (w.r.t. the classes we have assigned to them) such as the attacked services, their life time, behavior of attacking sources, and distribution of number of platforms and countries involved in the attack events. As a step further in analyzing the attack events, taking advantage of data mining techniques, we have proposed a method to group together related attack events into what we called *action set*. Studying those action sets reveals several interesting aspects of the threats. For instance, in the case of botnets, we have observed that some botnets last as long as 700 days. It is interesting to see that botnets adopt several strategies for renewing their zombie machines. Also, some botnets managed to survive for a very long period of time even though the renewal rate of their zombie machines is extremely high. We have also shown that we can deduce the type of attack tools employed by the zombie machines in botnets. Indeed, we have shown that there are several kind of advanced and sophisticated behaviors infected machines can have such as fingerprint, multi-attack vector, multi-headed. We have also observed that botnets frequently change their attack vectors in different attack campaigns. It is important to highlight that our solutions are easy to deploy and do not rely on any assumption about the communication protocol used by botnets. Our hope is to provide insights into the bigger picture of today's (and yesterday's) botnets activities. This kind of knowledge will help us to have a more precise perception on the current threats our computers are facing, and thus, to build a better countermeasure.

7.2 Future Work

We have shown that the analysis of attack events can indeed cast more light on the attack phenomena we observe. Given the first promising results, it would be interesting to carry out more works around the attack event concept. Indeed, there are at least three research directions that can follow these works :

1. We have observed that in starting from different viewpoints namely country and platform, we ended up identifying different sets of attack events. We believe that exploiting more dimensions may help us to identify more attack events. For instance, we could analyze the impact of time step granularity on the detection of attack events. Also, instead of splitting the attacks by country, we could split them by IP block...
2. We have shown that by analyzing the attack events we can understand more about the attack tools, the modus operandi of certain attack classes as well as

several other characteristics. The second research direction could be to apply a more systematical analysis framework to analyze the attack events and extract more knowledge from them. It is important to highlight the fact that the attack events generated as an output of our approach can be and, actually, have been used by other researchers in the context of their own work in this research direction [122, 123]

3. There are several open questions left in this work. For instance, as we mentioned earlier, we do not have a valid explanation for the existence of attack events of class *others*. The main reason may be that, with low interaction honeypot, we can not have a full understanding on the motivation of the attacks. The third research direction could be to use the external sources of information to enrich our knowledge about different attack phenomena we observe.

Publications

1. Pham, Van-Hau ;Dacier, Marc. Honeypot traces forensics : the observation view point matters. In *NSS 2009, 3rd International Conference on Network and System Security*, October 19-21, 2009, Gold Cost, Australia.
   Best paper award
2. Marc, Dacier ; Pham, Van-Hau ; Thonnard, Olivier. The WOMBAT Attack Attribution method : some results. In *Fifth International Conference on Information Systems Security (ICISS 2009)*, 14-18 December 2009, Kolkata, India
3. Pham, Van-Hau ;Dacier, Marc ;Urvoy-Keller, Guillaume ;En-Najjary, Taoufik. 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.
4. Leita, Corrado ; Pham, Van Hau ; Thonnard, Olivier ; Ramirez-Silva, Eduardo ; Pouget, Fabien ; Kirda, Engin ; Dacier, Marc. The leurre.com project : collecting internet threats information using a worldwide distributed honeynet. In *WISTDCS'08, 1st WOMBAT Workshop*, April 21st-22nd, 2008, Amsterdam, The Netherlands , pp 40-57
5. Pouget, Fabien ;Dacier, Marc ;Pham, Van Hau. Understanding threats : A prerequisite to enhance survivability of computing systems. In *International Journal of Critical Infrastructures*, Volume 4, N^{o}1-2, 2008, p 153-171
6. Alata, Eric ;Dacier, Marc ;Deswarte, Yves ;Kaaniche, Mohamed ;Kortchinsky, Kostya ;Nicomette, Vincente ;Pham, Van Hau ;Pouget, Fabien. Collection and analysis of attack data based on honeypots deployed on the Internet. In *QOP 2005, 1st Workshop on Quality of Protection (collocated with ESORICS and METRICS)*, September 15, 2005, Milan, Italy - Also published as Quality Of Protection, Security Measurements and Metrics, in *Springer Series : Advances in Information Security*, Volume 23, Gollmann, Dieter ; Massacci, Fabio ; Yautsiukhin, Artsiom (Eds.), 2006, XII, 197 p, ISBN : 0-387-29016-8
7. Alata, Eric ;Dacier, Marc ;Deswarte, Yves ;Kaâniche, Mohamed ;Kortchinsky, Kostya ;Nicomette, Vincente ;Pham, Van Hau ;Pouget, Fabien. CADHO : Collection and Analysis of Data from Honeypots. In *EDDC'05, 5th European Dependable Computing Conference*, April 20-22, 2005, Budapest, Hungary
8. Pouget, Fabien ;Dacier, Marc ;Pham, Van Hau. Leurre.com : on the advantages of deploying a large scale distributed honeypot platform. In *ECCE'05, E-Crime and Computer Conference*, 29-30th March 2005, Monaco

Bibliography

[1] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zait. Querying shapes of histories. In *the 21st Internal Conference on Very Large Databases*, pages 502–514, Zurich, Switzerland, Sept 11-15 1995.

[2] Eric Alata. *Observation, caracterisation et modelisation de processus dâĂŹattaques sur Internet.* PhD thesis, Institut National des Sciences Appliquees de Toulouse, 2007.

[3] Eric Alata, Marc Dacier, Yves Deswarte, Mohamed Kaaniche, Kostya Kortchinsky, Vincente Nicomette, Van Hau Pham, and Fabien Pouget. Collection and analysis of attack data based on honeypots deployed on the Internet. In *QOP 2005, 1st Workshop on Quality of Protection (collocated with ESORICS and METRICS), September 15, 2005, Milan, Italy - Also published as Quality Of Protection, Security Measurements and Metrics, Springer Series : Advances in Information Security , Volum*, Sep 2005.

[4] M. Allman, E. Blanton, V. Paxson, and S. Shenker. Fighting coordinated attackers with cross-organizational information sharing. In *Hotnets 2006*, 2006.

[5] Mark Allman, Vern Paxson, and Jeff Terrell. A brief history of scanning. In *IMC '07 : Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 77–82, New York, NY, USA, 2007. ACM.

[6] David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter : characterizing internet scam hosting infrastructure. In *SS'07 : Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–14, Berkeley, CA, USA, 2007. USENIX Association.

[7] Alberto Apostolico, Mary Ellen Bock, and Stefano Lonardi. Monotony of surprise and large-scale quest for unusual words. In *RECOMB '02 : Proceedings of the sixth annual international conference on Computational biology*, pages 22–31, New York, NY, USA, 2002. ACM.

[8] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The nepenthes platform : An efficient approach to collect malware. In *9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*, September 2006.

[9] Michael Bailey, Evan Cooke, Farnam Jahanian, Jose Nazario, and David Watson. The internet motion sensor : A distributed blackhole monitoring system. In *12th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, February 2005.

[10] Michael Bailey, Evan Cooke, Farnam Jahanian, Niels Provos, Karl Rosaen, and David Watson. Data reduction for the scalable automated analysis of distributed darknet traffic. In *IMC '05 : Proceedings of the 5th ACM SIGCOMM conference on Internet measurement*, pages 1–14, New York, NY, USA, 2005. ACM.

[11] Michael Bailey, Evan Cooke, Farnam Jahanian, David Watson, and Jose Nazario. The blaster worm : Then and now. *IEEE Security and Privacy*, 3(4) :26–31, 2005.

[12] Paul Barford and Vinod Yegneswaran. An inside look at botnets. *Advances in Information Security*, 27 :171–191, 2007.

[13] Alex Brodsky and Dmitry Brodsky. A distributed content independent method for spam detection. In *HotBots'07 : Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 3–3, Berkeley, CA, USA, 2007. USENIX Association.

[14] Caida Project. The ucsd network telescope. www.caida.org, 2007.

[15] CERT. Advisory CA-2003-20 W32/Blaster worm, August 2003.

[16] Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2) :188–228, 2002.

[17] Bee-Chung Chen, Vinod Yegneswaran, Paul Barford, and Raghu Ramakrishnan. Toward a query language for network attack data. In *ICDEW '06 : Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06)*, page 28, Washington, DC, USA, 2006. IEEE Computer Society.

[18] P T Chen, C S Laih, Fabien Pouget, and Marc Dacier. Comparative survey of local honeypot sensors to assist network forensics. In *SADFE'05, 1rst International Workshop on Sytematic Approaches to Digital Forensic Engineering, November 7-9, 2005, Taipei, Taiwan*, Nov 2005.

[19] Zesheng Chen and Chuanyi Ji. Optimal worm-scanning method using vulnerable-host distributions. *Int. J. Secur. Netw.*, 2(1/2) :71–80, 2007.

[20] Ken Chiang and Levi Lloyd. A case study of the rustock rootkit and spam bot. In *First Workshop on Hot Topics in Understanding Botnets*, 2007.

[21] Cisco. Netflow services solutions guide.

[22] Evan Cooke, Michael Bailey, Z. Morley Mao, David Watson, Farnam Jahanian, and Danny 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.

[23] Evan Cooke, Farnam Jahanian, and Danny McPherson. The zombie roundup : understanding, detecting, and disrupting botnets. In *SRUTI'05 : Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, pages 6–6, Berkeley, CA, USA, 2005. USENIX Association.

[24] INFOSEC Research Council. Hard problem list-2005. www.cyber.st.dhs.gov/docs/IRC_Hard_Problem_List.pdf, November 2005.

[25] Weidong Cui, Randy H. Katz, and Wai-tian Tan. Protocol-independent adaptive replay of application dialog. In *The 13th Annual Network and Distributed System Security Symposium (NDSS)*, February 2006.

[26] CWSandbox - Automated Behavior Analysis of Malware. www.cwsandbox.org, 2007.

[27] TEAM CYMRU. http ://www.team-cymru.org/.

[28] Marc Dacier, Fabien Pouget, and H. Debar. Attack processes found on the internet. In *NATO Symposium IST-041/RSY-013*, Toulouse, France, April 2004.

[29] Marc Dacier, Fabien Pouget, and Hervé Debar. Towards a better understanding of internet threats to enhance survivability. In *the International Infrastructure Survivability Workshop 2004 (IISW 04)*, 2004.

[30] Neil Daswani and Michael Stoppelman. The anatomy of clickbot.a. In *HotBots'07 : Proceedings of the First Workshop on Hot Topics in Understanding Botnets*, pages 11–11, Berkeley, CA, USA, 2007. USENIX Association.

[31] Texas A&M University Department of Statistics. Stat 30x : Statistical methods. http ://www.stat.tamu.edu/stat30x/notes/node41.html, 2008.

[32] DShield. Distributed intrusion detection system. www.dshield.org, 2007.

[33] Thomas Dubendorfer, Arno Wagner, Theus Hossmann, and Bernhard Plattner. Flow-level traffic analysis of the blaster and sobig worm outbreaks in an internet backbone. In *DIMVA 2005*, 2005.

[34] Valerie J. Easton and John H. McColl. Statistics glossary v1.1.

[35] Karim El Defrawy, Minas Gjoka, and Athina Markopoulou. Bottorrent : misusing bittorrent to launch ddos attacks. In *SRUTI'07 : Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet*, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association.

[36] EURECOM. Leurre.com, Eurecom Honeypot Project. http ://www.leurrecom.org/.

[37] F-Secure. F-secure virus descriptions : Nimda. http ://www.f-secure.com/v-descs/nimda.shtml, September 2001.

[38] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. *SIGMOD Rec.*, 23(2) :419–429, 1994.

[39] Jason Franklin, Adrian Perrig, Vern Paxson, and Stefan Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *CCS '07 : Proceedings of the 14th ACM conference on Computer and communications security*, pages 375–388, New York, NY, USA, 2007. ACM.

[40] Felix C. Freiling, Thorsten Holz, and Georg Wicherski. Botnet tracking : Exploring a root-cause methodology to prevent distributed denial-of-service attacks. In *Lecture Notes in Computer Science*, pages 319–335. Springer-Verlag GmbH, September 2005.

[41] Carrie Gates and John McHugh. The contact surface : A technique for exploring internet scale emergent behaviors. In Diego Zamboni, editor, *DIMVA*, volume 5137 of *Lecture Notes in Computer Science*, pages 228–246. Springer, 2008.

[42] P Geurts. Pattern extraction for time series classification. In *the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 115–127, Freiburg, Germany, Sep 3-7 2001.

[43] Jan Goebel and Thorsten Holz. Rishi : Identify bot contaminated hosts by irc nickname evaluation. In *Workshop on Hot Topics in Understanding Botnets 2007*, 2007.

[44] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-peer botnets : overview and case study. In *HotBots'07 : Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 1–1, Berkeley, CA, USA, 2007. USENIX Association.

[45] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer : Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *USENIX Security '08*, 2008.

[46] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. Bothunter : Detecting malware infection through ids-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium*, August 2007.

[47] Guofei Gu, Junjie Zhang, and Wenke Lee. Botsniffer : Detecting botnet command and control channels in network traffic. In *the 15th Annual Network and Distributed System Security Symposion*, 2008.

[48] John Heidemann, Yuri Pradkin, Ramesh Govindan, Christos Papadopoulos, Genevieve Bartlett, and Joseph Bannister. Census and survey of the visible internet. In *Proceedings of the ACM Internet Measurement Conference*, pages 169–182, Vouliagmeni, Greece, October 2008. ACM.

[49] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and detecting fast-flux service networks. In *NDSS 2008*, 2008.

[50] Thorsten Holz, Moritz Steiner, Frederic Dahl, Ernst Biersack, and Felix Freiling. Measurements and mitigation of peer-to-peer-based botnets : a case study on storm worm. In *LEET'08 : Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–9, Berkeley, CA, USA, 2008. USENIX Association.

[51] Nicholas Ianelli and Aaron Hackworth. Botnets as a vehicle for online crime. In *18th Annual FIRST Conference*, May 2007.

[52] Ari Juels, Sid Stamm, and Markus Jakobsson. Combating click fraud via premium clicks. In *SS'07 : Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–10, Berkeley, CA, USA, 2007. USENIX Association.

[53] Jaeyeon Jung and Emil Sit. An empirical study of spam traffic and the use of dns black lists. In *IMC '04 : Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 370–375, New York, NY, USA, 2004. ACM.

[54] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur W. Berger. Botz-4-sale : Surviving organized ddos attacks that mimic flash crowds. In *2nd Symposium on Networked Systems Design and Implementation (NSDI)*, May 2005.

[55] Jayanthkumar Kannan, Jaeyeon Jung, Vern Paxson, and Can Emre Koksal. Semi-automated discovery of application session structure. In *IMC '06 : Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 119–132, New York, NY, USA, 2006. ACM.

[56] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. Blinc : multilevel traffic classification in the dark. In *SIGCOMM '05 : Proceedings of the 2005 conference on Applications, technologies, architectures, and*

protocols for computer communications, pages 229–240, New York, NY, USA, 2005. ACM.

[57] Sachin Katti, Balachander Krishnamurthy, and Dina Katabi. Collaborating against common enemies. In *IMC '05 : Proceedings of the 5th ACM SIGCOMM conference on Internet measurement*, pages 1–14, New York, NY, USA, 2005. ACM.

[58] William Kent. A simple guide to five normal forms in relational database theory. *Commun. ACM*, 26(2) :120–125, 1983.

[59] E. Keogh and M Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *the 4th Int'l Conference on Knowledge Discovery and Data Mining*, pages 239–241, New York, Aug 27-31 1998.

[60] E. Keogh and M. Pazzani. Derivative dynamic time warping, 2001.

[61] Eamonn Keogh. Data mining and machine learning in time series databases.

[62] J. Kruskall and M. Liberman. The symmetric time warping problem : From continuous to discrete. In *Time Warps, String Edits and Macromolecules : The Theory and Practice of Sequence Comparison*, 1983.

[63] International Secure Systems Lab. Anubis : Analyzing unknown binaries. http ://anubis.iseclab.org/.

[64] R. J. Larsen and M. L. Marx. *An Introduction to Mathematical Statistics and Its Applications*. Prentice Hall, 2001.

[65] Corrado Leita, Marc Dacier, and Frédéric 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.

[66] Corrado Leita, Ken Mermoud, and Marc Dacier. Scriptgen : an automated script generation tool for honeyd. In *Proceedings of the 21st Annual Computer Security Applications Conference*, December 2005.

[67] Corrado Leita, Van Hau Pham, Olivier Thonnard, Eduardo Ramirez Silva, Fabien Pouget, Engin Kirda, and Marc Dacier. The leurre.com project : collecting internet threats information using a worldwide distributed honeynet. In *1st WOMBAT workshop, April 21st-22nd, Amsterdam, The Netherlands*, Apr 2008.

[68] Xin Li, Fang Bian, Hui Zhang, Christophe Diot, Ramesh Govindan, Wei Hong Hong, and Gianluca Lannaccone. Advanced indexing techniques for wide-area network monitoring. In *ICDEW '05 : Proceedings of the 21st International*

Conference on Data Engineering Workshops, page 1184, Washington, DC, USA, 2005. IEEE Computer Society.

[69] Zhichun Li, Anup Goyal, Yan Chen, and Vern Paxson. Automating analysis of large-scale botnet probing events. In *ACM Symposium on Information, Computer and Communications Security*, 2009.

[70] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD '03 : Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, New York, NY, USA, 2003. ACM Press.

[71] Maxmind. IP Geolocation and Online Fraud Prevention. www.maxmind.com.

[72] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2) :39–53, 2004.

[73] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the slammer worm. *IEEE Security and Privacy*, 1(4) :33–39, 2003.

[74] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring internet denial-of-service activity. *ACM Trans. Comput. Syst.*, 24(2) :115–139, 2006.

[75] David Moore, Colleen Shannon, and Jeffery Brown. Code-red : a case study on the spread and victims of an internet worm. In *Proceedings of the Internet Measurement Workshop (IMW)*, 2002.

[76] Mwcollect. The mwcollect alliance - http ://alliance.mwcollect.org.

[77] myNetWatchman. mynetwatchman - network intrusion detection and reporting. http ://www.mynetwatchman.com.

[78] Naoum Naoumov and Keith Ross. Exploiting p2p systems for ddos attacks. In *InfoScale '06 : Proceedings of the 1st international conference on Scalable information systems*, page 47, New York, NY, USA, 2006. ACM.

[79] J. Nazario, J. Anderson, R. Wash, and C. Connelly. The future of internet worms. In *Black Hat 2001*, USA, 2001.

[80] Jose Nazario. Phoneyc : A virtual client honeypot. In *LEET2009*, 2009.

[81] Pang, Yegneswaran, Barford, Paxson, and Peterson. Characteristics of background radiation. In *Proceedings of the 4th ACM SIGCOMM conference on the Internet Measurement*, 2004.

[82] Emanuele Passerini, Roberto Paleari, Lorenzo Martignoni, and Danilo Bruschi. Fluxor : detecting and monitoring fast- flux service networks. In *DIMVA 2008*, 2008.

[83] Vern Paxson. Experiences with countering internet attacks.

[84] Vern Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31(3) :38–47, 2001.

[85] Van-Hau Pham, Marc Dacier, Guillaume Urvoy Keller, and Taoufik 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.

[86] Kin pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.

[87] Fabien Pouget. *Distributed System of Honeypots Sensors : Discrimination and Correlative Analysis of Attack Processes*. PhD thesis, Institut Eurecom, 2006.

[88] Fabien Pouget and Marc Dacier. Honeypot-based forensics. In *AusCERT2004, AusCERT Asia Pacific Information technology Security Conference 2004, 23rd - 27th May 2004, Brisbane, Australia*, May 2004.

[89] Fabien Pouget, Marc Dacier, and H. Debar. Honeypot-based forensics. In *Proceedings of AusCERT Asia Pacific Information Technology Security Conference 2004*, Brisbane, Australia, May 2004.

[90] Fabien Pouget, Marc Dacier, and Hervé Debar. Honeypots, a practical mean to validate malicious fault assumptions. In *PRDC'04, 10th International symposium Pacific Rim dependable computing Conference, March 3-5, 2004, Tahiti, French Polynesia*, Mar 2004.

[91] Fabien Pouget, Marc Dacier, and Van Hau 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.

[92] Fabien Pouget, Marc Dacier, and Van Hau Pham. Leurre.com : on the advantages of deploying a large scale distributed honeypot platform. In *ECCE'05, E-Crime and Computer Conference, 29-30th March 2005, Monaco*, Mar 2005.

[93] Fabien Pouget, Guillaume Urvoy Keller, and Marc Dacier. Time signatures to detect multi-headed stealthy attack tools. In *18th Annual FIRST Conference, June 25-30, 2006, Baltimore, USA*, Jun 2006.

[94] Netgeo Product. Home page of the netgeo company at http ://www.netgeo.com/.

[95] TCPDUMP Project. Home page of the tcpdump project at http ://www.tcpdump.org/.

[96] The Honeynet Project. Honeywall. https ://projects.honeynet.org/honeywall/.

[97] The Honeynet Project. Honeywall. https ://projects.honeynet.org/honeywall/.

[98] Niels Provos. A virtual honeypot framework. In *Proceedings of the 12th USENIX Security Symposium*, pages 1–14, August 2004.

[99] Niels Provos, Panayiotis Mavrommatis, Moheeb Abu Rajab, and Fabian Monrose. All your iframes point to us. In *USENIX Security '08*, 2008.

[100] Moheeb Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *ACM SIGCOMM/USENIX Internet Measurement Conference*, October 2006.

[101] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. *SIGCOMM Comput. Commun. Rev.*, 36(4) :291–302, 2006.

[102] Anirudh Ramachandran, Nick Feamster, and David Dagon. Revealing botnet membership using dnsbl counter-intelligence. In *SRUTI'06 : Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet*, pages 8–8, Berkeley, CA, USA, 2006. USENIX Association.

[103] James Riordan, Diego Zamboni, and Yann Duponchel. Building and deploying billy goat, a worm detection system. In *Proceedings of the 18th Annual FIRST Conference*, 2006.

[104] S. Salvador and P. Chan. Fastdtw : Toward accurate dynamic time warping in linear time and space. In *3rd Wkshp. on Mining Temporal and Sequential Data, ACM KDD '04*, 2004.

[105] Stuart E. Schechter and Michael D. Smith. Access for sale a new class of worm. In *WORM'03*, 2003.

[106] ALMODE Security. Home page of disco at at http ://www.altmode.com/disco/.

[107] Daniel R. Simon, Sharad Agarwal, and David A. Maltz. As-based accountability as a cost-effective ddos defense. In *HotBots'07 : Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 9–9, Berkeley, CA, USA, 2007. USENIX Association.

[108] Dawn Song, David Brumley, Heng Yin, Juan Caballero, Ivan Jager, Min Gyung Kang, Zhenkai Liang, James Newsome, Pongsin Poosankam, and Prateek Saxena. Bitblaze : A new approach to computer security via binary analysis. In *ICISS '08 : Proceedings of the 4th International Conference on Information Systems Security*, pages 1–25, Berlin, Heidelberg, 2008. Springer-Verlag.

[109] Lance Spitzner. *Honeypots : Tracking Hackers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[110] Stuart Staniford, David Moore, Vern Paxson, and Nicholas Weaver. The top speed of flash worms. In *WORM '04 : Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 33–42, New York, NY, USA, 2004. ACM Press.

[111] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to own the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149–167, Berkeley, CA, USA, 2002. USENIX Association.

[112] Günther Starnberger, Christopher Krügel, and Engin Kirda. Overbot - A botnet protocol based on Kademlia. In *SecureComm 2008, 4th International Conference on Security and Privacy in Communication Networks, September 22-25th 2008, Istanbul, Turkey*, Sep 2008.

[113] Joe Stewart. Sinit p2p trojan analysis. http ://www.secureworks.com/research/threats/sinit/, December 2003.

[114] Joe Stewart. Bobax trojan analysis. http ://www.secureworks.com/research/threats/bobax, May 2004.

[115] Joe Stewart. Phatbot trojan analysis. http ://www.secureworks.com/research/threats/sinit/, March 2004.

[116] Joe Stewart. Spamthru trojan analysis. http ://www.secureworks.com/research/threats/spamthru, 2006.

[117] W. Timothy Strayer, Robert Walsh, Carl Livadas, and David Lapsley. Detecting botnets with tight command and control. *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 195–202, Nov. 2006.

[118] Andy Sudduth. The what, why, and how of the 1988 internet worm. http ://snowplow.org/tom/worm/worm.html.

[119] Symantec. Deepsight early warning services.

[120] Symantec. Symantec global internet security threat report trends for 2008, April 2009.

[121] Olivier Thonnard and Marc Dacier. A framework for attack patterns' discovery in honeynet data. *DFRWS 2008, 8th Digital Forensics Research Conference, August 11- 13, 2008, Baltimore, USA*, 2008.

[122] Olivier Thonnard, Wim Mees, and Marc Dacier. Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making. In *KDDâĂŽ09, 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on CyberSecurity and Intelligence Informatics, June 28th - July 1st, 2009, Paris, France*, Dec 2009.

[123] Olivier Thonnard, Olivier Thonnard, and Marc 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.

[124] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.*, 363(1) :28–42, 2006.

[125] N. Vanderavero, X. Brouckaert, O. Bonaventure, and B. Le Charlier. The HoneyTank : a scalable approach to collect malicious Internet traffic. In *IISW04*, 2004.

[126] Shobha Venkataraman, Subhabrata Sen, Oliver Spatscheck, Patrick Haffner, and Dawn Song. Exploiting network structure for proactive spam mitigation. In *SS'07 : Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–18, Berkeley, CA, USA, 2007. USENIX Association.

[127] Ping Wang, Sherri Sparks, and Cliff C. Zou. An advanced hybrid peer-to-peer botnet. In *HotBots'07 : Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 2–2, Berkeley, CA, USA, 2007. USENIX Association.

[128] Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. A taxonomy of computer worms. In *WORM '03 : Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 11–18, New York, NY, USA, 2003. ACM Press.

[129] V. Yegneswaran, P. Barford, and V. Paxson. Using honeynets for internet situational awareness. In *the ACM/USENIX Hotnets IV*, 2005.

[130] Vinod Yegneswaran, Paul Barford, and Dave Plonka. On the design and use of internet sinks for network abuse monitoring. In *In Proceedings of the 7 th International Symposium on Recent Advances in Intrusion Detection (RAID*, pages 146–165, 2004.

[131] Vinod Yegneswaran, Paul Barford, and Johannes Ullrich. Internet intrusions : global characteristics and prevalence. *SIGMETRICS Perform. Eval. Rev.*, 31(1) :138–147, 2003.

[132] Byoung-Kee Yi and Christos Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB '00 : Proceedings of the 26th International Conference on Very Large Data Bases*, pages 385–394, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[133] Yi-Min. Automated web patrol with strider honeymonkeys : Finding web sites that exploit browser vulnerabilities. In *NDSS2006*, 2006.

[134] M. Zalewski and W. Stearns. Passive OS Fingerprinting Tool.

[135] Jian Zhang, Phillip Porras, and Johannes Ullrich. Highly predictive blacklisting. In *Usenix Security 2008*, 2008.

[136] J. Zhuge, T. Holz, X. Han, J. Guo, and W. Zou. Characterizing the irc-based botnet phenomenon. Technical report, Peking University & University of Mannheim, 2007.

[137] Jianwei Zhuge, Thorsten Holz, Xinhui Han, Chengyu Song, and Wei Zou. *Collecting Autonomous Spreading Malware Using High-interaction Honeypots*, chapter Collecting Autonomous Spreading Malware Using High-Interaction Honeypots, pages 438–451. Springer Berlin / Heidelberg, 2007.

[138] Jacob Zimmermann, Andrew Clark, George Mohay, Fabien Pouget, and Marc Dacier. The use of packet inter-arrival times for investigating unsolicited internet traffic. page 89, 2005.

[139] Cliff C. Zou, Don Towsley, Weibo Gong, and Songlin Cai. Routing worm : A fast, selective attack worm based on ip address information. In *PADS '05 : Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, pages 199–206, Washington, DC, USA, 2005. IEEE Computer Society.