Classification et caractérisation non supervisée des attaques vers des pots de miel

Richard Turc (rturc@laas.fr)*
Philippe Owezarski (owe@laas.fr) *

Résumé :
L'observation des réseaux et de leur trafic est importante pour l'estimation du risque dans l'Internet et par voie de conséquence la protection des systèmes informatiques. Cette observation se fait notamment par le biais de sondes de métrologie et de pots de miel. Mais l'analyse des quantités de données collectées, et la caractérisation des anomalies et attaques contenues dans ces traces restent des tâches aussi compliquées que fastidieuses, faites par des experts, et sont par conséquent lentes et coûteuses. Dans cet article, nous présentons une méthode non-supervisée pour classifier et caractériser les anomalies contenues dans le trafic des réseaux de pots de miel. Cette méthode autonome n'utilise ni signature, ni apprentissage, ni trafic documenté, ni connaissances préalables en matière de sécurité, ce qui constitue un progrés majeur en direction de l'apparition de systèmes de sécurité autonomes. Cet article montre finalement comment il est possible à partir des résultats de caractérisation des anomalies d'inférer les règles de filtrages qui pourront automatiquement servir à configurer les filtres des composants du réseau ou des pare-feux.

Mots Clés : trafic de pots de miel, classification d'anomalies, apprentissage non supervisé, sécurité autonome

1 Introduction

L'observation des réseaux et de leur trafic est importante dans la protection des systèmes informatiques. Plusieurs méthodes d'observation sont possibles. Par exemple, l'utilisation de sondes mises à l'entrée des réseaux permet d'observer tous les flux qui entrent et qui sortent. Une autre méthode consiste à installer dans un réseau des machines appelées "pots de miel". Ces machines offrent des services plus ou moins émulés (en fonction du niveau d'interaction souhaité avec l'attaquant) et permettent d'observer l'utilisation d'un service par un attaquant. Ainsi, les données collectées permettent aux

∗. CNRS, LAAS, 7 Avenue du Colonel Roche, F-31400 Toulouse, France
Univ. de Toulouse, LAAS, F-31400 Toulouse, France

experts, d'une part de mieux comprendre les faiblesses des systèmes informatiques, mais aussi de collecter des informations sur les activités des pirates qui permettront d'analyser leurs méthodes, objectifs et stratégies d'attaques. L'estimation et l'analyse du risque induit par les activités illicites de l'Internet sont des éléments essentiels pour que les experts en sécurité puissent concevoir et développer des systèmes de défense et de protection adaptés et efficaces, en relation avec le risque ambiant.

De ce fait, la caractérisation et la classification des anomalies et attaques actuelles rencontrées dans l'Internet est une tâche aussi compliquée que fastidieuse, faite par des experts. Elle est par conséquent lente et coûteuse. La principale difficulté liée à l'identification et à l'analyse des différentes classes de trafics illicites dans les réseaux est leur capacité à évoluer, à s'amplifier et à se renouveler. L'autonomie du processus de classification et de caractérisation nous semble donc cruciale pour la mise en place de systèmes faciles à déployer et utiliser. Les systèmes de classification modernes doivent donc proposer des capacités de classification et de caractérisation ne nécessitant pas d'intervention humaine et capables de s'adapter automatiquement à l'évolution de toutes les composantes du trafic, normales et anormales (et bien sûr cet article se focalise sur les composantes anormales). Nous proposons pour cela d'utiliser des techniques d'apprentissage non-supervisé ne requérant aucune intervention humaine en mode opérationnel. Dans cet article, nous présentons une méthode non-supervisée pour classifier et caractériser les anomalies contenues dans du trafic collectés par des de pots de miel. Cette méthode n'utilise ni signature, ni apprentissage, ni trafic documenté, ce qui constitue un progrés majeur en direction de l'apparition de systèmes de sécurité autonomes. Notre approche utilise des techniques de *clustering* (partitionnement) robustes, notamment le *sub-space clustering* (partitionnement en sous-espace), le *density-based clustering* (partitionnement basé sur la densité), et l'accumulation de preuves pour classifier des ensembles de flux en classes de trafic et construire des signatures associées, aisément compréhensibles.

Le fonctionnement de cet algorithme sera illustré sur l'analyse de traces de trafic à destination ou issues de pots de miel. Ces traces ont été collectées à l'université du Maryland, au format netflow [BCH+10].

2 Agrégation des données

Les données dont nous disposons sont complexes. Elles contiennent pour chaque flux de nombreuses informations : adresses source et destination, ports source et destination, protocole transporté, les flags des protocoles, etc. Au total, nous avons défini un ensemble d'une cinquantaine d'attributs simples ou composés [MCLO11]. Cela revient donc pour un algorithme de clustering à réaliser des analyses dans des espaces de très grandes dimensions, ce pour quoi les algorithmes de clustering manquent de performance et de précision. Le principe du sub-space clustering consiste à projeter le grand espace d'étude initial dans des espaces de dimensions plus réduites pour gagner en efficacité et en ro-

bustesse. D'autre part, les différentes attaques contenues dans le trafic des pots de miel peuvent avoir des caractéristiques variables notamment en termes d'origine et de destination. Aussi, avant toute analyse, nous agrègeons les données dans des *truncs* de trafic de taille variable afin d'analyser des classes de trafic illicites, de compositions, de sources et de destinations variables.

Soit E l'ensemble des flux à analyser. Nous allons agréger plusieurs fois tous ces flux. Chaque agrégation se fait sur un intervalle de temps et avec un autre au moins des attributs. Nous avons empiriquement mis en évidence 8 niveaux d'agrégations importants l_i. Ils ont pour attribut d'agrégation : l'adresse IP source $(l_1 : saddr)$, l'adresse IP destination $(l_2 : daddr)$, le préfixe réseau source $(l_{3,4,5}) : saddr/24, /16, /8)$, le préfixe réseau destination $(l_{6,7,8}) : daddr/24, /16, /8)$.

Soit $Y = \{y_1, ...y_n\}$ l'ensemble des flux agrégés selon un au moins des niveaux d'agrégation précédents l_i. Chaque flux $y_f \in Y$ est décrit par un ensemble de A attributs de trafic sur lesquels est faite l'analyse. $x_f = (x_f(1), ..., x_f(A)) \in \mathbb{R}^A$ est le vecteur des attributs décrivant le flux de trafic y_f, et $X = \{x_1, ..., x_n\}$ la matrice des attributs, appelée aussi espace d'attributs. Une fois l'espace d'attributs construit, un algorithme de partitionnement est utilisé afin d'identifier les classes de trafic au sein de Y.

3 Sub-space Clustering

Le clustering est l'étape qui va permettre de regrouper les flux qui ont des caractéristiques communes et qui vont former les différents clusters correspondant aux différentes classes dans le trafic analysé. Pour que l'algorithme fonctionne, il faut définir une fonction de comparaison entre deux flux. Or nos flux sont caractérisés par plus de cinquante attributs. La définition de la fonction est essentielle.

Pour répondre aux problèmes de dimensionnement de données, de comparaison et d'évolution, le partitionnement en sous espace, dit *Sub-space Clustering* en anglais, a été proposé [MCLO11]. Le principe est de projeter l'espace des flux identifiés par ses cinquante attributs sur un ensemble exhaustif d'espaces de tailles plus réduites (pour des raisons de lisibilité des représentations graphiques, nous utilisons dans cet article des espaces à 2 dimensions correspondant à des projections sur 2 attributs). L'algorithme de partitionnement de données qui a été sélectionné est DBSCAN [EKSX96]. Cet algorithme identifie les clusters par densité de points (ici un point correspond à un flux), et présente l'avantage de ne pas avoir à indiquer au préalable le nombre de classes de trafic (clusters) qui doivent être identifiées dans le trafic total analysé. Dans le contexte de sous espaces à deux dimensions, l'algorithme DBSCAN regroupe dans un même cluster les flux $y_f \in Y$ qui ont deux attributs dont les valeurs sont proches. L'algorithme retourne alors l'ensemble des couples d'attributs $K_{(n,m)} = \{(x_{1n}, x_{1m}), ..., (x_{fn}, x_{fm})\}$ des points qui constituent les différents clusters.

Par exemple, la figure 1 montre les différents groupes du sous espace $nSyn/nPkts$

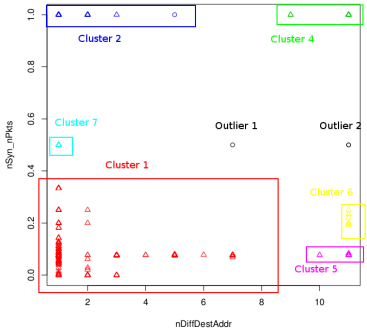

FIGURE 1: Les groupes (clusters) pour le sous espace (nSyn/nPkts, nDiffDestAddr) avec une agrégation par IP source et par intervalle de 1 minute

et $nDiffDestAddr$. Nous identifions 7 groupes possédant des caractéristiques différentes. Dans l'exemple, le cluster 2 regroupe l'ensemble des flux contenant seulement des paquets tcp avec le flag *syn* actif et ciblant une à cinq machines distinctes.

Une fois les clusters dans les sous espaces générés par l'algorithme DBSCAN, nous les regroupons en classes de trafic. Une classe de trafic consiste en un ensemble de flux similaires. Pour les regrouper, il faut d'abord connaître la similitude entre les groupes des différents espaces. Le résultat est mis dans la matrice carrée S de taille N qui correspond au nombre total de clusters à comparer. La fonction donnant le taux de similitude entre deux clusters, est la suivante :

$$CS(C_d, C_e) = \frac{|C_d| \cap |C_e|}{max(|C_d|, |C_e|)} \tag{1}$$

où $|C_n|$ est le cardinal du cluster C_n. On obtient donc la matrice de similitude S de l'ensemble de clusters C :

$$N = |C|$$

$$\forall (i,j), 1 \leq i \leq N \wedge \forall j, 1 \leq j \leq N \Rightarrow [S]_{ij} = CS(C_i, C_j) \tag{2}$$

Ensuite, pour chaque élément de S, nous voulons savoir s'il existe un niveau de similitude suffisant qui indiquerait un lien entre deux clusters de sous espaces différents. Il existe quand la valeur de la similitude est supérieure à un seuil λ. $if [S]_{ij} > \lambda \Rightarrow C_i \sim C_j$. La matrice générée peut être alors considérée comme une matrice adjacente et être transformée en graphe, comme sur la Figure 2. Les sommets du graphe représentent les clusters des différents sous espaces, tandis que les arcs existent s'il y a un lien entre deux

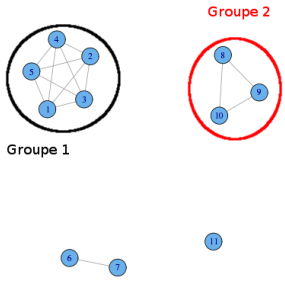

FIGURE 2: Graphe de similitude avec regroupement

groupes de deux sous espaces. Avec ce graphe, il est plus facile d'identifier les clusters des différents sous espaces qui sont liés.

En effet, il suffit de récupérer les cliques maximales, par exemple, le cluster 1 (ou groupe 1) et le cluster 2 de la Figure 2. Comme cela est présenté dans [MCLO11], chaque clique de groupes de sous espaces est considérée comme une classe de trafic. De plus, comme nous observons le trafic attaquant des pots de miel, les classes de trafic récupérées sont illégitimes. Dans la suite, on qualifiera ces différentes classes d'anomalies.

L'algorithme de sub-space clustering doit être fait pour chaque niveau d'agrégation. Les classes illégitimes ainsi trouvées dans chaque sous-espace et pour chaque niveau d'agrégation ont ensuite été regroupées pour mettre en évidence leur nombre dans le trafic original initial. Toutefois, nous allons montrer dans la suite que certains recoupements entre ces anomalies existent et ne sont pas mis en évidence par l'accumulation de preuves que nous venons d'utiliser. Pour corriger l'algorithme, nous allons présenter un nouveau mécanisme de corrélation des anomalies.

4 Corrélation des classes de trafic illégitimes

Nous avons réussi à détecter plusieurs classes de trafic illégitimes mais celles-ci se situent dans des niveaux d'agrégations différents. Nous savons que des classes se trouvant dans des niveaux différents peuvent être liées. Deux classes de trafic de deux niveaux d'agrégations différents sont liées par exemple quand les flux composant les classes proviennent des mêmes sources et parviennent aux mêmes cibles. Il faut donc pouvoir lier les classes illégitimes correspondant à une seule anomalie. La corrélation des classes de trafic illégitimes répond à cet objectif ; elle sert à déterminer la ressemblance entre les classes de chaque niveau pour essayer de les regrouper en un seul ensemble d'anomalies.

Pour évaluer la ressemblance entre deux classes illégitimes qui sont sur des niveaux différents, nous utilisons une fonction de comparaison entre ces deux classes. Notre méthode de comparaison utilise une comparaison d'adresses IP [MCLO11]. Pour utiliser

month	day	hour	min	saddr	daddr	nPkts	nBytes	nSyn/nPkts
02	01	02	4	192.168.0.1, 192.168.0.2	172.16.4.16, 172.16.4.21	54	4726	0.1481481
02	01	03	40	192.168.0.1, 192.168.0.3	172.16.4.16, 172.16.4.21	54	4869	0.1481481
02	01	18	05	192.168.0.1, 192.168.0.3	172.16.4.16, 172.16.4.21	53	3996	0.0754717
02	01	19	55	192.168.0.1, 192.168.0.2	172.16.4.16, 172.16.4.21	54	4545	0.1481481

TABLE 1: Premier exemple d'une classe de trafic illégitime (les adresses IP ont été anonymisées)

month	day	hour	min	saddr	daddr	nPkts	nBytes	nSyn/nPkts
04	10	01	30	192.168.0.1, 192.168.0.2	172.16.4.16, 172.16.4.21	5	20	1
04	11	01	30	192.168.0.1, 192.168.0.3	172.16.4.16, 172.16.4.21	4	160	1
04	12	01	30	192.168.0.1, 192.168.0.2	172.16.4.16, 172.16.4.21	5	20	1
04	13	01	30	192.168.0.1	172.16.4.16, 172.16.4.21	4	160	1

TABLE 2: Second exemple d'une classe de trafic illégitime (les adresses IP ont été anonymisées)

cette méthode, il faut récupérer les adresses IP sources et destinations des classes de trafic a_1 et de a_2. Ensuite, il faut comparer, en utilisant la fonction (3), les adresses IP sources entre elles puis les adresses destinations.

$$Sim_{@}(@_1, @_2) = \frac{|@_1 \cap @_2|}{max(|@_1|, |@_2|)} \tag{3}$$

où $@_n$ est un ensemble d'adresses IP et $|@_n|$ son cardinal.

Il existe une ressemblance entre deux classes illégitimes si l'équation (4) est vraie. $tTSAddrSims$ est un seuil à définir.

$$Sim_{src}(@_1, @_2) \ > \ tTSAddrSim \ \wedge \ Sim_{dest}(@_1, @_2) \ > \ tTSAddrSims \tag{4}$$

Le problème de cette méthode est qu'elle ne prend pas en compte le temps. Or, comme il se peut que dans la même classe, il y ait des empreintes qui ont été émises avec plusieurs mois d'écart, on ne peut pas considérer les adresses IP seules.

Par exemple, les classes de trafic illégitimes du tableau 1 et 2 sont totalement différentes de par leur comportement. Pour la première classe, le nombre de paquets émis dans chaque flux vaut environ 54, alors que pour la deuxième, il vaut environ 4. De plus, les temps d'émission ont deux mois d'écart. Les deux seuls liens qu'il y ait entre les deux classes est l'adresse IP des émetteurs (saddr) et l'adresse IP des cibles (daddr). Dans ce cas, le taux de ressemblance entre les deux classes de trafic aurait été de 66%, ce qui est très élevé, et il y aurait eu un lien entre ces deux classes, alors qu'il est fort probable que les logiciels malveillants infectants les machines émettrices ne soient plus les mêmes.

C'est pour cela que la notion de temps a été ajoutée dans la corrélation.

Considérons deux classes de trafic illégitimes a_1 et a_2. T_1 et T_2 sont les ensembles d'intervalles de temps dans lesquels il y a des empreintes de a_1 et de a_2. $@_1^n$ et $@_2^n$ sont les ensembles d'adresses IP source ou destination des classes a_1 et a_2 pour l'intervalle $t_n \in T_1$. La fonction de similarité est définie par :

$$SimTime_{@} : \sum_{t \in T_1} \frac{Sim_{@}(@_1^t, @_2^t)}{max(|@_1^t|, |@_2^t|)} \tag{5}$$

Mais cette méthode a un défaut. Quand il y a des décalages de temps entre deux classes, la similitude vaut quasiment zéro alors que deux classes peuvent avoir une grande ressemblance pour d'autres attributs. L'ensemble des adresses IP des classes d'anomalies a_1 et a_2 sont donc utilisées. Les adresses sont alors indépendantes du temps.

$$SimGlobal_{@} : \frac{Sim_{@}(@_1, @_2)}{max(|@_1|, |@_2|)} \tag{6}$$

Nous obtenons la fonction de similitude, qui est définie dans (7).

$$\begin{aligned} SimAnomalies(a1, a2) : SimTime_{Src}(a_1, a_2) > \lambda_1 \wedge SimTime_{Dest}(a_1, a_2) > \lambda_1 \\ \wedge\, SimGlobal_{Src}(a_1, a_2) > \lambda_2 \wedge SimGlobal_{Dest}(a_1, a_2) > \lambda_2 \end{aligned} \tag{7}$$

Le résultat de cette fonction est un booléen qui vaut vrai si a_1 et a_2 sont similaires.

Reprenons les exemples des classes de trafic des tableaux 1 et 2. Nous ne devons pas trouver de lien entre ces deux classes du fait de la disparité du temps. Commençons par prendre les adresses IP de la première classe pour le 1er février à 2 heures 40. Nous avons les adresses IP 192.168.0.1 et 192.168.0.2. Maintenant pour la deuxième classe, nous avons un ensemble vide. La ressemblance entre les deux ensembles est nulle. Si on continue avec les autres dates et avec la même méthode, nous obtenons toujours une valeur nulle et le résultat le sera aussi. La nouvelle fonction ne trouve donc pas de lien entre ces deux classes de trafic illégitimes.

En revanche, avec cette nouvelle fonction, il faut sélectionner les valeurs de seuils appropriées. Si les seuils sont mal choisis, la fonction ne trouvera aucun lien entre les classes de trafic. Avec un λ_1 faible, à chaque fois qu'il y a une petite similitude temporelle entre deux classes illégitimes, ce sera alors la valeur de la similitude globale qui déterminera s'il y a une ressemblance entre les deux classes. Tandis que si λ_1 est élevé, il y aura une ressemblance entre deux classes lorsque celles-ci émettront quasiment en même temps, vers les mêmes cibles et les mêmes sources. Ceci est très contraignant et empêche toute existence de décalages temporels. La similitude globale aura alors moins d'importance. A partir de résultats obtenus empiriquement, il est recommandé de prendre pour λ_1 un seuil assez faible, entre 10% et 30%, et pour λ_2 un seuil supérieur à 25%.

Les classes de trafic illégitimes qui apparaissaient disjointes suite à la phase de sub-space clustering peuvent donc, grâce à cette méthode, être regroupées car il est mis en évidence qu'elles correspondent à une seule et même anomalie.

5 Caractérisation des classes d'anomalies

A ce stade, le trafic global issu ou à destination des pots de miel a été décomposé en classes qui indiquent pour chacune d'elles des comportements différents. Les techniques de classification du trafic Internet doivent donc alors identifier le type de trafic, les applications, ou le type d'anomalies ou d'attaques que représentent chacun de ces clusters. Les techniques de classification les plus avancées du moment - des techniques semi-supervisées - utilisent alors des signatures propres à chacune de ces familles, issues soit d'une connaissance préalable, soit d'une étape d'apprentissage sur un trafic connu, dont les composantes applicatives auront déjà été labelisées. C'est naturellement une obligation par rapport aux objectifs de classification, mais qui reste un frein dans l'optique d'une méthode totalement autonome.

Nous n'avons pas ce type de contrainte étant donné la nature du trafic que nous nous devons classifier : du trafic illicite. Ainsi, il n'est pas indispensable de parfaitement identifier le type d'anomalie (ou d'attaque) ni de la nommer. En effet, toutes les classes de trafic mise en évidence par l'algorithme de sub-space clustering et d'accumulation d'évidences sont des anomalies ou des attaques. De fait, le traitement à appliquer consistera in fine à éliminer ce trafic après en avoir extrait toute connaissance nécessaire pour par exemple estimer un risque dans l'Internet.

Nous proposons pour cela de générer automatiquement des règles caractérisant les classes d'anomalies. Avec ces règles, un opérateur est capable de comprendre les propriétés des anomalies et d'en déduire les contre-mesures à appliquer.

Les règles sont de deux types : les règles absolues et les règles relatives [MCLO11].

Une règle absolue se caractérise par l'apparition d'une valeur dominante pour certains attributs des flux de l'anomalie. Il suffit alors d'extraire à partir des flux de Y définissant l'anomalie les attributs ayant la même valeur. Soit a un attribut de l'anomalie et a_f sa valeur pour $y_f \in Y_g \subset Y$. Un attribut est défini absolu quand :

$$\{\forall y_f \in Y_g \subset Y, a_f == \lambda\} \tag{8}$$

La règle absolue s'écrira donc :

$$\{a == \lambda\} \tag{9}$$

Par exemple, sur la Figure 3, les clusters 2 et 3 ont la valeur de nSyn/nPkts toujours à 1. On en déduit donc la règle absolue $\{nSyn/nPkts == 1\}$.

Par contre, pour définir une règle relative, nous devons séparer, dans un sous-espace, l'ensemble des flux composant un cluster des autres clusters ou outliers. S'il y a plusieurs

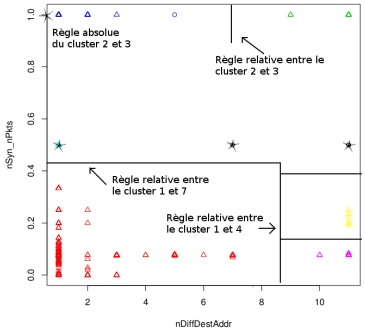

FIGURE 3: Les différentes règles pour le sous espace (nSyn/nPkts, nDiffDestAddr) avec une agrégation par IP source et par intervalle de 1 minute

règles, nous prenons celle qui est la plus discriminante. La règle relative, s'écrira alors $\{a < \lambda\}$ ou $\{a > \lambda\}$.

Par exemple, imaginons qu'une anomalie soit composée du groupe de sous espace 1 (cluster 1) qui est présent sur la Figure 3. Il existe une règle relative entre le cluster 1 et le cluster 4. Pour la générer, nous avons pris les clusters 1 et 4 et nous avons cherché la valeur médiane entre les deux. Avec cette valeur, nous pouvons créer la règle relative suivante $\{nDiffDestAddr < 9\}$ comme caractérisant l'anomalie associée au cluster 1. Par contre, l'anomalie associée au cluster 4, sera caractérisée par la règle $\{nDiffDestAddr > 9\}$.

Une règle est donc composée d'un ensemble de contraintes qui s'appliquent sur un attribut a d'un flux $y_f \in Y_g \subset Y$. Il y a quatre types de contraintes possibles :

— $FR(Y_g, a) = \{\forall y_f \in Y_g \subset Y, a == \lambda\}$
— $FR(Y_g, a) = \{\forall y_f \in Y_g \subset Y, a < \lambda\}$
— $FR(Y_g, a) = \{\forall y_f \in Y_g \subset Y, a > \lambda\}$
— $FR(Y_g, a) = \{\forall y_f \in Y_g \subset Y, \lambda_1 < a < \lambda_2\}$

Nous avons déjà vu que les anomalies, provenant de plusieurs niveaux d'agrégation (l_1, l_2, ...), étaient corrélées entre elles. Cela signifie qu'une anomalie peut avoir des règles différentes en fonction du niveau d'agrégation. Or nous voulons un minimum de règles pour caractériser les anomalies. Nous devons donc faire une génération de règles pour chaque niveau et les fusionner lorsqu'elles se superposent. Cette fusion relève de l'arithmétique simple. Ainsi si une règle r_1 indique $attribut < \lambda_1$ et un règle r_2 indique $attribut < \lambda_2$ avec $\lambda_1 < \lambda_2$ alors la règle fusionnée r est $attribut < \lambda_1$.

Quand toutes les contraintes sont fusionnées, la règle générale est obtenue. Cette règle décrit le comportement de l'anomalie. Par exemple, pour un scan réseau qui va

impliquer, entre autre, la proportion de paquets SYN, la règle automatiquement produite est $(\mathrm{nSrcs} == 1) \wedge (\mathrm{nDsts} > \lambda_1) \wedge (\mathrm{nSYN/nPkts} > \lambda_2)$, où λ_1 et λ_2 sont deux seuils obtenus en séparant des clusters à mi-distance (via la médiane).

6 Conclusion

Cet article a présenté un algorithme non-supervisé de classification des anomalies qui présente plusieurs avantages par rapport à l'ensemble des travaux existants : (i) il fonctionne d'une manière complètement non-supervisée, ce qui signifie qu'il peut être associé à n'importe quel système de monitoring et être utilisé directement, sans configuration ou connaissance préalable. (ii) Il combine des techniques de clustering robuste afin d'éviter les problèmes habituels des algorithmes de partitionnement, e.g. : la sensibilité à l'initialisation, l'indication préalable du nombre de clusters ou la perturbation des résultats par l'utilisation d'attributs non pertinents. (iii) Il construit automatiquement des signatures simples et concises qui caractérisent les attaques qui peuvent alors être utilisées dans un outil de sécurité. (iv) Il est conçu pour fonctionner en temps-réel en permettant d'exploiter le parallélisme de notre approche de clustering.

Cet algorithme ouvre ainsi de nouvelle perspectives pour permettre une analyse du risque dans l'Internet - faite à partir des traces de trafic de pots de miels - et installer automatiquement les règles de filtrage associées sur des pare-feu ou en utilisant les règles de filtrage des équipements du réseau.

7 Remerciements

Les auteurs tiennent à remercier sincèrement Michel Cukier et Bertrand Sobesto pour avoir mis à notre disposition les traces de trafic collectées sur les pots de miel de l'université du Maryland.

Références

[BCH+10] Robin Berthier, Michel Cukier, Matti Hiltunen, Dave Kormann, Gregg Vesonder, and Dan Sheleheda. Nfsight : netflow-based network awareness tool. In *Proceedings of the 24th international conference on Large installation system administration (LISA'10)*, 2010.

[EKSX96] M. Ester, H.-P Kriegel, J. Sander, and X. Xu. A dendity-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings 2nd International Conference on knowledge Discovery and Data mining*, pages 226–231, 1996.

[MCLO11] Johan Mazel, Pedro Casas, Yann Labit, and Philippe Owezarski. Sub-space clustering, interclustering results association & anomaly correlation for un-

supervised network anomaly detection. In *7th International Conference on Network and Service Management (CNSM 2011), CNSM'11*, october 2011.