Détection d'anomalies réseau par apprentissage non supervisé

José Márcio Martins da Cruz

Détection d'anomalies réseau par apprentissage non supervisé

José-Marcio Martins da Cruz
MINES Paristech
Centre de calcul et systèmes d'information
60, bd Saint Michel
75 006 Paris

Résumé

Le fonctionnement habituel des dispositifs de détection d'incidents réseau ou de filtrage de flux est basé sur des règles construites à partir de connaissances à priori : des accès légitimes ou, l'inverse, les accès interdits ou malveillants.

Et pourtant, des techniques d'apprentissage automatique existent mais elles restent dans le domaine de la recherche académique ou, peut-être, dans des systèmes propriétaires et fermés.

Les expérimentations académiques sont, et c'est normal, évaluées avec des jeux de données standardisés. Nous avons construit un prototype dans le but de voir ce qu'un dispositif de ce genre peut nous apporter dans un environnement typique d'un organisme d'enseignement/recherche (université ou école).

Les résultats que nous avons obtenu jusqu'à maintenant montrent que, déjà avec des heuristiques simples, le résultat positif le plus immédiat est une meilleure connaissance du fonctionnement du réseau et que des anomalies triviales (scans et similaires) sont facilement identifiables.

Mots clefs

Sécurité, incidents, détection d'anomalies, anomalies réseau, apprentissage non supervisé

1 Introduction

La sécurité réseau est devenue une préoccupation depuis les années 80 et en particulier avec le *Ver Morris* [1] qui profitait d'une vulnérabilité connue de *sendmail*. Ce qui était juste un « jeu innocent » pour son auteur (un étudiant) lui a couté une condamnation pénale et a déclenché la création du premier CERT (*Computer Emergency Response Team*). Au fil du temps, ce qui étaient des jeux ou des challenges sont devenus des activités avec des enjeux économiques, politiques, sécuritaires et humains puisque les attaques sont, maintenant, en rapport avec des activités d'espionnage, terrorisme ou banditisme.

Les systèmes informatiques sont, depuis ces années là, devenus beaucoup plus complexes de telle manière que la surface d'attaque est bien plus importante et les vulnérabilités plus subtiles et difficiles à découvrir et à protéger. Il n'est plus étonnant de voir qu'un système a été pénétré et que l'intrusion n'a été découverte que plusieurs mois ou années après.

Ainsi, pour bien tenir son réseau il faut, d'une part, avoir des outils de filtrage (parefeu) correctement configurés et à jour et, d'autre part, bien connaître le fonctionnement normal de son réseau et mettre en place des outils de surveillance (IDS et NIDS [2]) capables de détecter tout fonctionnement anormal ou suspect.

1. http://histoire-internet.vincaria.net/post/histoire/internet/1988/Ver-Morris
2. IDS pour *Intrusion Detection System* et NIDS pour *Network Intrusion Detection System*

Ces systèmes de filtrage et de monitoring fonctionnent, quasiment tous, avec des règles construites à partir de connaissances à priori, en deux modes (ou un mixte des deux) :

— **détection d'abus** (*misuse detection*) - on cherche à détecter tout comportement présent dans une liste de comportements interdits ou malveillants ;

— **détection d'anomalies** (*anomaly detection*) - au contraire, on détecte tout comportement non présent dans une liste de comportements considérés légitimes.

Le premier cas exige des actualisations en permanence, vu la rapidité d'apparition de nouvelles vulnérabilités et modes d'attaque, et n'est pas capable de détecter des attaques encore inconnues. Le deuxième mode détecte des abus inconnus mais génère des erreurs dans les deux sens : des fausses alertes et laisse passer des attaques dissimulées dans des fonctionnements considérés normaux. Pour une plus ample discussion sur ces approches voir le guide du NIST [1] ou le chapitre 2 de Ghorbani [2].

Idéalement, il serait souhaitable d'avoir un système capable d'observer le trafic et détecter des anomalies, même celles inconnues. Ces souhaits émergent naturellement à une époque où les techniques de traitement massif de données et d'intelligence artificielle - apprentissage automatique (*machine learning*) et fouille de données (*data mining*) - sont des sujets très à la mode.

Des travaux de recherche existent depuis plus de vingt ans mais, à notre connaissance, ils restent encore dans le domaine de la recherche académique ou, alors, comme outil d'aide à la construction de règles et signatures.

En fait, la détection d'anomalies, ou d'intrusions, dans un réseau est un problème particulier : parmi les difficultés, il s'agit de détecter des événements rares pour lesquels le coût des erreurs est assez élevé. Il y a aussi, très certainement, une dose de méfiance envers des systèmes que l'on ne maîtrise pas parfaitement. Sommer et Paxson [3] proposent une discussion intéressante, suggérant que le problème doit avoir une approche plus pragmatique et des objectifs plus précis et réalistes.

Dans le but de voir ce que peut apporter ce type de technique dans un environnement interne d'une école, nous avons mis en place un prototype basé sur de l'apprentissage non supervisé (*clustering*), permettant de regrouper les paquets selon leur « ressemblance ». Les données analysées sont issus d'une sonde « *netflow* » en routeur d'entrée de site.

2 Présentation

Cette section précise le contexte de notre expérimentation dans le contexte général des dispositifs de détection d'événements de sécurité.

2.1 La détection d'événements réseau

Notre choix a été d'observer les trames en circulation sur le réseau (*network based*) grâce à une sonde de type `netflow`, placée soit sur le routeur d'entrée de site, soit sur le cœur de réseau. Ce type de sonde fournit des informations de synthèse sur les paquets vus : les adresses et numéros de port (source et destination), le protocole, la taille des paquets et les drapeaux. Par exemple :

```
1442582713   1.2.3.4:49668    8.8.8.85:23     tcp   1   60 SYN
1442582714   1.2.3.4:38347    8.8.8.104:23    tcp   1   60 SYN
1442582714   1.2.3.4:38490    8.8.8.111:23    tcp   1   60 SYN
```

D'autres types de sonde, telles `tcpdump`, peuvent accéder au contenu des paquets et effectuer des analyses plus poussées.

Certains événements peuvent être parfaitement identifiés puisque les informations sont suffisantes : des *scans* de port (exemple ci-dessus) puisqu'on verra une suite de paquets de la même source extérieure allant vers le même port de plusieurs destinations internes. D'autres événements, par exemple, un serveur interne de résolution DNS (résolveur) faisant une quantité importante de requêtes vers le même serveur DNS externe peut suggérer une attaque sur le DNS (faille Kaminsky, p. ex.) mais peut aussi concerner un serveur de messagerie qui consulte une liste noire extérieure. Autrement dit, dans certains cas ce sont les conséquences qui sont observées et pas les causes : des observations *indirectes*.

2.2 Apprentissage Automatique et travaux similaires

Le processus d'apprentissage automatique consiste à créer un modèle, grâce à des exemples initialement disponibles, permettant d'« identifier » des objets nouveaux. En *apprentissage supervisé*, une étiquette est associée à chaque exemple (*bon/mauvais*, *petit/moyen/grand*, ...), ce qui n'est pas le cas en *apprentissage non supervisé* où l'on va se contenter de regrouper (*clustering*) les objets selon leur « ressemblance ». Les techniques d'apprentissage automatique utilisées dans ces travaux sont décrites dans, p. ex., le livre de Vipin Kumar[4].

Depuis longtemps, plusieurs approches de détection d'intrusion dans un réseau basées aussi bien sur ces deux modes d'apprentissage ont été proposées : une typologie est présentée par Kishor et all[5].

Parmi les nombreuses solutions basées sur l'apprentissage non supervisé, citons en trois. Barbara et all (ADAM) [6] en 2001 initialise le processus par la création d'un modèle de trafic pendant une période sans anomalies (apprentissage). La détection de déviations se fait par comparaison des composantes d'un flot avec celles d'un trafic normal. Dokas et all [7] tient compte du fait que les anomalies constituent des « événements rares ». Mazel [8] en 2011 (thèse à l'INSA-Toulouse) propose l'analyse de séries temporelles construites à partir de paramètres globaux de trafic pour détecter des périodes de fonctionnement anormal pour ensuite chercher les catégories minoritaires de trafic à l'intérieur de la période. Il n'y a pas de construction de modèle initial, mais les anomalies ne sont détectables qu'après une délai suffisant pour avoir suffisamment de points dans la série temporelle. On utilise, dans les trois cas, des fenêtres temporelles de taille fixe.

Notre prototype est, en partie, inspiré par les deux premiers travaux, tout en prennant quelques idées des travaux de Mazel [8].

Une mise en œuvre typique contiendrait les modules suivants (cf figure 1) :

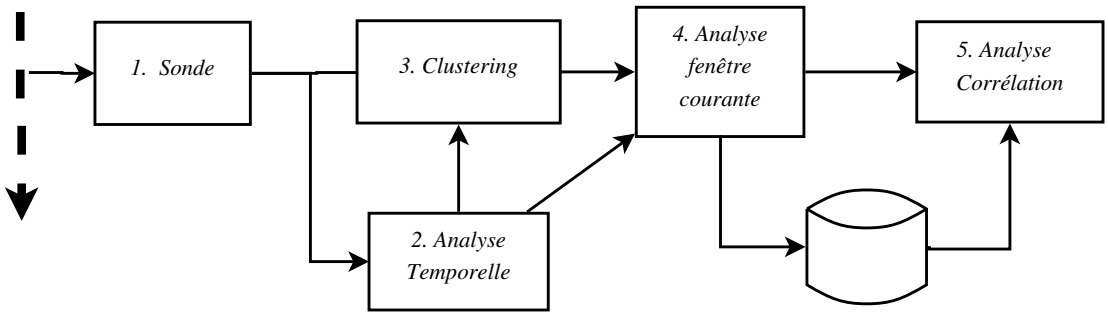

Figure 1 - Exemple d'architecture de système de détection

1. une sonde pour la collecte d'informations sur les trames ;

2. un module d'extraction et suivi temporel des indicateurs globaux de trafic, pouvant modifier les paramètres des modules suivants ;
3. un module d'extraction (*clustering*) des composantes de trafic de la fenêtre en cours ;
4. un module de traitement des composantes de trafic de la fenêtre en cours avec sélection de celles les plus significatives ;
5. un module d'analyse temporelle des composantes et détection d'anomalies.

Dans l'état actuel notre prototype intègre les modules 1. et 3. ainsi qu'une partie des modules 2. et 4.. Ces modules suffisent pour évaluer la potentialité de la méthode et poser les bases de réflexion sur les techniques envisageables pour le cinquième module.

3 Mise en œuvre

Notre prototype est capable de recevoir, en temps réel, les informations *netflow* envoyées par un routeur et de les traiter, en mode différé. Nous avons choisi de traiter séparément les flux entrant et sortant, même si parfois leur corrélation pourrait être utile.

Le premier module, la *sonde*, est un serveur *UDP*, écrit en *C*, qui reçoit les paquets *netflow* envoyés par un routeur et enregistre, dans un format texte simple, les informations à traiter. Les autres modules ont été écrits en *perl*, langage pratique pour le prototypage de ce type de traitement. La mise en production dans un site plus important exigera, très certainement, le passage à un langage compilé.

Le principe du *clustering* est le regroupement des objets selon leur similarité. Vu que les informations (adresses et numéros de port) ne sont pas des valeurs numériques, la similarité consiste à tenir compte de l'égalité ou pas pour le protocole et numéro de port, et plage pour les adresses. Cela implique, naturellement, que le *clustering* doit être fait à plusieurs échelles, définies dans le tableau 1.

				Source		Destinataire		
ID	Mode	Mask	Proto	Addr	Port	Addr	Port	Exemple d'anomalie
1	1 :N	= ;32 ;* ;24 ;=	=	/32	*	/24	=	Scan réseau, propagation de ver
2	1 :N	= ;32 ;* ;24 ;*	=	/32	*	/24	*	Scan exhaustif lent
3	1 :1	= ;32 ;* ;32 ;*	=	/32	*	/32	*	Scan de ports
4	1 :1	= ;32 ;= ;32 ;=	=	/32	=	/32	=	Connexion atypique
5	1 :1	= ;32 ;* ;32 ;=	=	/32	*	/32	=	DoS
6	N :1	= ;24 ;* ;32 ;=	=	/24	*	/32	=	DDoS
7	N :1	= ;16 ;* ;32 ;=	=	/16	*	/32	=	DDoS

Tableau 1 - Critères de similarité pour agrégation de paquets

Chaque ligne définit un critère d'agrégation. Prenons, comme exemple, la première ligne : on agrège dans un même groupe les paquets ayant un même protocole (*tcp* ou *udp*), la même adresse IP source (*/32*), quel que soit le port (*), les destinataires dans un même réseau (*/24*) et un même numéro de port (=). Bien évidement, un *cluster* peut se trouver inclus dans un cluster de plus haut niveau et une procédure d'élimination doit faire partie de l'analyse faite dans le module 4.

L'analyse se fait par l'évaluation des attributs du *cluster* : taille, nombre de paquets `SYN`, nombre d'adresses source et destination, nombre de ports, nombre de paquets ainsi que des ratios entre ces valeurs. Mazel [8] utilise ces attributs pour agréger à nouveau le trafic et identifier les anomalies (*outliers*).

L'autre aspect important est la taille de la fenêtre d'observation. En fait, la taille de la fenêtre est un compromis : pas trop importante pour que le système puisse être extensible (*scalability*) et pas trop petite pour

que son contenu soit significatif. Mazel [8] a trouvé, empiriquement une valeur idéale de l'ordre de 15 s. Sans contredire son choix, le nôtre, empirique lui aussi, porte sur une fenêtre qui doit être entre 2 et 5 minutes. En effet, une fenêtre plus longue exige des ressources plus importantes (mémoire et CPU) mais peut simplifier les traitements postérieurs puisque son contenu devient plus informatif et peut, dans beaucoup de cas, préciser déjà la nature de chaque communication. Nous reviendrons sur ce point.

4 Résultats et Discussion

Nous avons, donc, créé un prototype capable d'analyser et agréger, à plusieurs échelles, les paquets selon leur ressemblance avec les critères de la Table 1, sur une fenêtre temporelle de taille fixe (5 minutes). Le prototype a été appliqué aux données collectées par une sonde du type *netflow* placée sur le routeur d'entrée du site de l'école à Paris.

Malgré la simplicité du prototype, les événements intéressants sont bien mis en évidence. Les Listings 1 et 2 correspondent, respectivement, à une recherche de machines avec le port SSH ouvert (*scan*) et à un robot qui s'acharne sur le *MX* du domaine pour distribuer du *spam*.

```
mask : =;32;*;24;= : tcp  1.2.3.6  *    7.7.7.0/24   22
  SRC         :  addr      1 - ports  10 (1210, ...)
  DST         :  addr     10 - ports   1 (22)
  FLAGS
        SYN   :       10
  Bytes       :     3360
  Pkts        :       10
```

Listing 1: Scan réseau - port 22 (ssh) sur fenêtre de 5 minutes

```
mask : =;32;*;32;= : tcp  1.2.3.7  *   8.8.8.196  25
  SRC         :  addr      1 - ports  431 (1026, ...)
  DST         :  addr      1 - ports    1 (25)
  FLAGS
    ACK   :   454 - FIN  :   452 - PUSH :   453 - SYN  :   454
  Bytes       :    96990
  Pkts        :     2266
```

Listing 2: Acharnement (spam) sur serveur SMTP

Remarque : Le comptage des drapeaux dans les paquets n'est, malheureusement, pas correct, mais une limite inférieure. En effet, Les paquets *netflow* sont, parfois, déjà partiellement agrégés par connexion et *netflow* n'indique que si un drapeau apparait dans une agrégation mais pas le nombre d'occurrences. Néanmoins, on observe surtout l'importance du drapeau SYN, toujours élevé dans la plupart des anomalies.

Nous ne présentons pas ici, mais on identifie tout aussi clairement les différentes connexions correspondantes au fonctionnement normal du réseau.

4.1 Discussion

Nous avons utilisé une taille de fenêtre temporelle assez importante (5 min), bien plus importante que celle utilisée par Mazel [8] (15 s). Si bien qu'il ne semble pas évident de pouvoir définir une taille de

fenêtre idéale, à cause de la diversité des attaques possibles, il semble qu'une taille de l'ordre de la minute semble un bon choix de façon à pouvoir capturer suffisamment d'information à l'intérieur de la fenêtre d'échantillonnage. Néanmoins, cette taille peut-être problématique dans des sites plus importants que le nôtre. Pour mémoire, les données utilisées par Mazel [8] en provenance de MAWI [9] (une sonde dans un backbone) sur une durée de 15 minutes sont de l'ordre de 3 fois plus importantes que le trafic chez nous sur une journée entière. Ce sont deux contextes différents.

Listing 3 et 4 sont deux exemples d'anomalies difficiles, voir impossible, à identifier avec des fenêtres petites. Listing 3 correspond au dialogue entre un poste infecté par le Cheval de Troie Strictor et un poste de commandement. Mis à part le port inhabituel (9507), il n'y a pas d'autre moyen de détection puisque le poste infecté ne contacte le serveur que quelques fois par jour et n'impacte pas les paramètres globaux de trafic.

```
t+0  8.8.8.221:53856    1.2.3.14:80         tcp 5 718 ACK,PUSH,SYN,FIN
t+0  1.2.3.14:80        8.8.8.221:53856     tcp 5 888 ACK,PUSH,SYN,FIN
...
t+14 8.8.8.221:51585    1.2.3.14:9507       tcp 4 671 ACK,PUSH,SYN
t+14 1.2.3.14:9507      8.8.8.221:51585     tcp 3 292 ACK,PUSH,SYN
t+59 8.8.8.221:51585    1.2.3.14:9507       tcp 2 104 ACK,FIN
t+59 1.2.3.14:9507      8.8.8.221:51585     tcp 1  52 ACK,FIN
```

Listing 3: Paquets échangés par un ordinateur infecté par le Cheval de Troie *Strictor*

Listing 4 correspond à un *scan* exhaustif très lent : ports de destination très diversifiés et intervalle très long entre deux paquets (moyenne de 800 paquets par jour). Il s'agit d'un *scan* qui s'étend sur plus d'un an, lancé par une dizaine de sources placées dans des réseaux différents pour rester inaperçu.

```
t       1.2.3.5:61819    8.8.8.9:21        tcp  1   40 SYN
t+133   1.2.3.5:34680    7.7.7.81:2480     tcp  1   40 SYN
t+256   1.2.3.5:44738    7.7.7.22:6379     tcp  1   40 SYN
t+550   1.2.3.5:20012    8.8.8.17:8443     tcp  1   40 SYN
t+729   1.2.3.5:11748    7.7.7.117:111     tcp  1   40 SYN
t+755   1.2.3.5:34680    8.8.8.174:8080    tcp  1   40 SYN
t+816   1.2.3.5:23183    8.8.8.81:37777    tcp  1   40 SYN
t+846   1.2.3.5:1959     7.7.7.7:789       tcp  1   40 SYN
t+864   1.2.3.5:63030    7.7.7.48:44818    udp  1   52 ACK
t+881   1.2.3.5:31632    8.8.8.103:9943    tcp  1   40 SYN
...
```

Listing 4: Scan exhaustif lent pendant des nombreux mois

Un sous-produit de ce système est une représentation des résultats sous la forme de graphe où les arrêtes sont construites par les flots agrégés et les sommets sont les noeuds (serveurs, postes de travail, ...). Bref, il s'agit d'utiliser ceci pour améliorer la connaissance de ce qui se passe à l'intérieur de notre réseau.

Malgré les bons résultats obtenus il ne faut pas être naïf : le niveau de trafic traité dans notre site est de l'ordre de 300 fois plus faible que celui de *MAWILab* [9] (une liaison entre universités américaines et japonaises). Dans ce contexte, où les anomalies seraient « noyées dans la masse », un algorithme aussi trivial ne suffira surement pas.

5 Conclusions

Les résultats obtenus jusqu'à maintenant montrent que, avec des heuristiques simples, le résultat positif le plus immédiat est une meilleure connaissance du fonctionnement du réseau et que des anomalies triviales (*scans* et similaires) sont facilement identifiables.

Ces résultats sont très satisfaisants pour un projet dont la seule prétention était de comprendre une technologie et son potentiel.

Par la suite, je vais surement continuer sur cette voie pour développer les briques manquantes et explorer des algorithmes d'apprentissage pas aussi triviaux.

Bibliographie

[1] Karen Scarfone et Peter Nell. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. NIST - National Institute of Standards and Technology, SP 800-94 - revision 1 édition, 2012.

[2] A. A. Ghorbani et W. Lu et M. Tavallaee. *Network Intrusion Detection and Prevention*. Springer-Verlag US, 2010.

[3] R. Sommer et V. Paxson. Outside the closed world : On using machine learning for network intrusion detection. Dans *IEEE Symposium on Security and Privacy*, pages 305–316, May 2010.

[4] P.-N. Tan et M. Steinback et V. Kumar. *Introduction to Data Mining*. Pearson Addison-Wesley, 2006.

[5] S. K. Wagh et V. K. Pachghare et S. R. Kolhe. Survey on intrusion detection system using machine learning techniques. *International Journal of Computer Applications*, 78(16) :30–37, September 2013.

[6] D. Barbará et J. Couto et S. Jajodia et L. Popyack et N. Wu. ADAM : Detecting intrusions by data mining. Dans *Proceedings of the IEEE Workshop on Information Assurance and Security*, pages 11–16, 2001.

[7] P. Dokas et L. Ertoz et V. Kumar et A. Lazarevic et J. Srivastava et P. Tan. Data mining for network intrusion detection. Dans *Proc. NSF Workshop on Next Generation Data Mining*, pages 15–24, 2002.

[8] Johan Mazel. *Unsupervised network anomaly detection*. Theses, INSA de Toulouse, Décembre 2011.

[9] R. Fontugne et P. Borgnat et P. Abry et K. Fukuda. MAWILab : Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. Dans *ACM CoNEXT '10*, Philadelphia, PA, December 2010.