Algorithme de détection d'attaques de type « SYN Flooding »

B. BENMAMMAR, C. LÉVY-LEDUC, F. ROUEFF

GET / Télécom Paris CNRS LTCI – 46, Rue Barrault - 75634 Paris Cedex 13

{benmamma,levyledu,roueff}@enst.fr

Résumé – Nous proposons un algorithme de détection d'anomalies dans le trafic Internet générées par une attaque de déni de service liée au protocole TCP : le « TCP/SYN Flooding ». Celui-ci permet de traiter une quantité très importante de données, détecte les anomalies en temps réel et identifie les adresses IP source et destination impliquées. Notre approche consiste à réduire la taille des données auxquelles on s'intéresse grâce une étape de filtrage et à analyser celles-ci grâce à une méthode statistique de détection de ruptures fondée sur un test de rang pour données censurées.

Abstract – We propose an intrusion detection system to detect anomalies in the Internet traffic caused by Denial of Service attacks such as TCP/SYN Flooding. Our approach can be used to treat a large amount of data, provides an on-line anomalies detection as well as the source and destination IP addresses at stake. Our method consists in a data reduction stage followed by a change-point detection method using a rank test for censored data.

1. Introduction

Le développement d'une politique de sécurité dans les réseaux IP est actuellement une véritable préoccupation à la fois pour les opérateurs de télécommunications, les entreprises et les collectivités locales. En effet, une intrusion dans un réseau peut avoir des effets dévastateurs pour l'organisme concerné. L'un des protocoles réseau les plus utilisés est le TCP/IP mais il n'est malheureusement pas sécurisé. C'est pour cela que l'on cherche à mettre en place des systèmes de détection d'intrusions. Le terme « intrusions » fait ici référence à des activités anormales ou suspectes et à des attaques sur un équipement du réseau. On peut trouver différents types d'attaques, la plus répandue sur Internet étant le « TCP/SYN Flooding » [1]. Il s'agit d'une attaque réseau par saturation (déni de service) exploitant la faiblesse intrinsèque du protocole TCP consistant en l'envoi massif de demandes de synchronisation (envoi de paquets SYN).

Plusieurs études se sont intéressées au problème de la détection d'attaques de type SYN flooding. On peut distinguer notamment des travaux présentant des méthodes pour réduire la taille des données utilisant des agrégations aléatoires ou sketches [2] et des algorithmes de détection de ruptures tels que le CUSUM [3], [4] et [5].

Ici, nous développons un algorithme pour détecter des attaques de type SYN Flooding que nous avons appliqué à des données produites par France Télécom dans le cadre du projet ANR-RNRT OSCAR. Le trafic en un point du réseau y est représenté sous la forme d'une succession de flots selon le format NetFlow incluant les instants de début et fin de flot, les adresses IP source et destination ainsi que le nombre de paquets transmis. Plus précisément, on étudie une trace qui dure approximativement 3 jours.

Nous proposons une approche originale pour réduire la taille des données tout en appliquant un traitement séquentiel à celles-ci. Ce traitement utilise une méthode statistique de détection de ruptures adaptée à la présence de données censurées développée dans [6]. Notre algorithme fournit en temps réel les instants où les attaques se produisent ainsi que les adresses IP concernées (attaquantes et attaquées).

2. Description de l'algorithme

Notre objectif est de détecter la présence d'un déni de service distribué (DDoS) de type « TCP/SYN flooding ». Pour cela, une démarche naturelle consisterait à suivre l'évolution temporelle du trafic à destination d'une adresse IP donnée et ceci pour chaque adresse IP destination apparaissant dans la trace. Mais, ceci s'avère trop coûteux en terme de temps de calcul puisque le fichier de données auquel on s'est intéressé contient plus de 90 millions de flots et met en jeu 3 millions d'adresses IP destination différentes, ce nombre augmentant de façon linéaire en fonction du temps pendant la durée de la trace. Pour contourner ce problème, nous proposons l'algorithme décrit ci-dessous.

Étape 1 : Filtrage par records

On segmente la trace en intervalles de temps de taille Δ s et sur chacun d'entre eux, on détermine et on classe M machines dans l'ordre décroissant du nombre de paquets SYN reçus. Cet ensemble de M machines sera noté Top M

dans la suite. On choisit Δ de telle sorte que le classement effectué soit véritablement sélectif.

Étape 2 : Création de séries temporelles censurées

On suit l'évolution temporelle des adresses IP destination ayant fait partie au moins une fois des ntop machines les plus sollicitées ($1 \le \text{ntop} \le M$), ceci réduisant de façon importante le nombre de séries temporelles à considérer dans la suite. Ainsi construites, ces séries temporelles comportent des données manquantes. En effet, il est possible qu'une adresse IP sélectionnée ne fasse pas partie des M machines les plus sollicitées en permanence. Dans ce cas, le nombre de paquets SYN qu'elle reçoit est censuré par celui de la M-ième machine la plus sollicitée.

Étape 3 : Test de détection de ruptures

Dans [6], un test statistique de détection de ruptures en présence de données censurées est proposé ainsi qu'une façon de calculer les p-valeurs associées. Il s'agit d'un test de rang non-paramétrique inspiré par le test de rang de Wilcoxon. Nous appliquons ce test sur une fenêtre d'observation de taille $P \times \Delta$ s à chacune des séries temporelles créées à l'étape précédente. Après avoir identifié les adresses IP ayant subi un changement de régime éventuel, on oublie les séries temporelles créées à l'Étape 2 lorsque l'on change de fenêtre d'observation. Grâce aux deux étapes précédentes, on ne suit finalement qu'au plus ntop $\times P$ séries temporelles.

Décrivons à présent plus précisément le test que nous effectuons ainsi que les données auxquelles nous l'appliquons. Pour chaque adresse IP destination (IPD) et pour chaque fenêtre d'observation, nous définissons $Y_t = (X_t , \delta_t)$ tel que :

$$X_t = \begin{cases} \text{Nombre de paquets SYN reçus par IPD pendant un intervalle de temps de taille } \Delta \textbf{ si IPD} \in \text{ Top M,} \\ \text{Nombre de paquets SYN reçus par la M-ième machine dans le Top M } \textbf{sinon,} \end{cases}$$

$$\delta_t = \begin{cases} 1 & \textbf{si} \quad \text{IPD} \in \text{ Top M,} \\ 0 & \textbf{sinon.} \end{cases}$$

On teste H_0 : « les $\{Y_t\}_{1 \le t \le P}$ sont des variables aléatoires iid » contre H_1 : « il existe r tel que : Y_1 , …, Y_r et Y_{r+1} , …, Y_P ont des lois différentes ».
Pour chaque i, j dans $\{1,..., P\}$, définissons :

- $H_{i,j} = I(X_i > X_j ,\ \delta_i = 1) - I(X_i < X_j ,\ \delta_j = 1)$.
- $U_i = \sum_{j=1}^{P} H_{i,j}$, $i = 1, \ldots, P$.
- $S_P = \max_{1 \le k \le P} \left| \sum_{i=1}^{k} U_i \right| \Big/ \left(\sum_{i=1}^{P} U_i^2 \right)^{1/2}$.

S_P est utilisée comme statistique de test. En effet, sous H_0, on a : $S_P \xrightarrow{D} S_\infty$, où $S_\infty = \sup_{0<t<1} |B(t)|$ et B est un pont Brownien.

La p-valeur associée à cette statistique de test est donnée par $Pval\ (S_p)$ où pour b > 0 (voir [6]) :

$$Pval\ (b) = P\ (S_\infty > b) = 2 \sum_{j=1}^{\infty} (-1)^{j-1} e^{-2 j^2 b^2} .$$

3. Résultats de l'expérimentation

3.1 Performances statistiques

Dans notre expérimentation, nous avons retenu la valeur Δ=10 s après avoir essayé différentes valeurs de Δ (5 s, 10 s, 20 s et 60 s), M=10 et P=100. Ces valeurs ont été choisies afin d'une part de pouvoir prendre une décision quant à la présence d'une éventuelle attaque toutes les 15 minutes environ ($P \times \Delta$ s) et d'autre part d'avoir suffisamment d'observations pour assurer la pertinence du test.

La Figure 1a ci-dessous représente le nombre de séries temporelles obtenues à l'Étape 2 à l'intérieur d'une fenêtre d'observation ainsi que son évolution au cours du temps lorsque ntop=1.

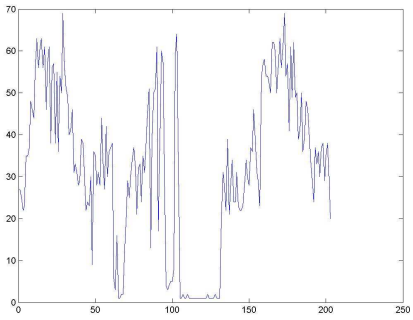

FIG. 1a : Nombre de séries temporelles (ntop=1)

La Figure 1b représente l'histogramme normalisé des p-valeurs associées au test décrit dans l'Étape 3 effectué sur les séries temporelles précédentes et son évolution au cours du temps.

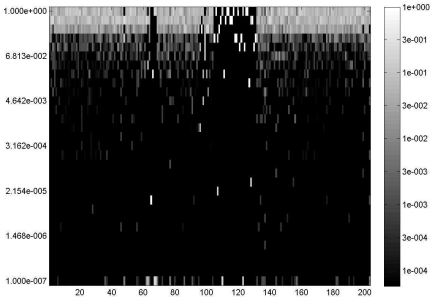

FIG. 1b : p-valeurs du test associé (ntop=1)

On remarque que prendre une p-valeur inférieure à 0.0001 permet de localiser des attaques aux instants où se produisent des pics d'activité mis en évidence dans la Figure 2. Celle-ci représente l'évolution temporelle de la machine ayant reçu le plus grand nombre de paquets SYN dans chaque intervalle de taille Δ s. Les pics d'activité sont

associés à des adresses IP que nous noterons dans la suite : IP1, IP2, IP3 et IP4.

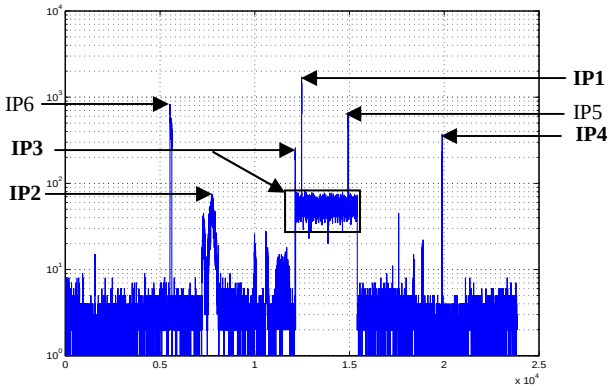

FIG. 2 : Série temporelle du « max »

Les Figures 3a, 4a, 5a et 6a représentent les séries temporelles censurées pour les adresses IP précédentes. Quant aux Figures 3b, 4b, 5b et 6b, elles représentent les p-valeurs correspondantes associées au test décrit dans l'Étape 3.

On remarque à nouveau que le test de rang censuré détecte correctement les changements de régimes dans les séries temporelles étudiées pour des p-valeurs inférieures à 0.0001. Ces changements de régime correspondent, comme attendu, aux pics d'activité de la Figure 2, notre méthode fournissant ainsi une localisation précise des phases d'attaques potentielles.

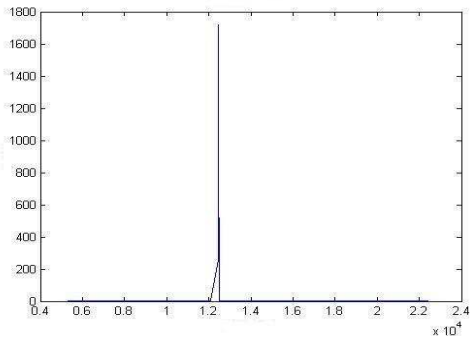

FIG. 3a : Série temporelle censurée de l'IP1

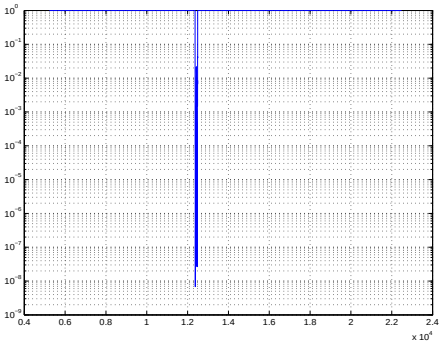

FIG. 3b : p-valeurs du test associé (IP1)

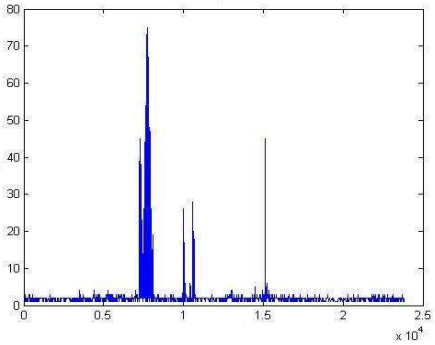

FIG. 4a : Série temporelle censurée de l'IP2

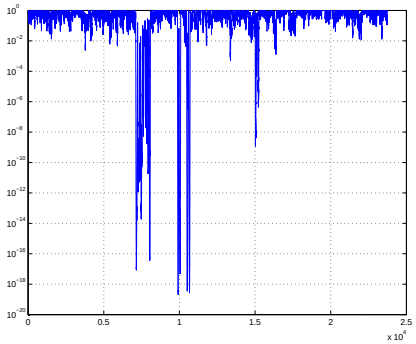

FIG. 4b : p-valeurs du test associé (IP2)

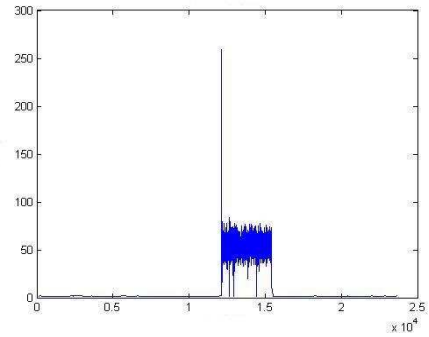

FIG. 5a : Série temporelle censurée de l'IP3

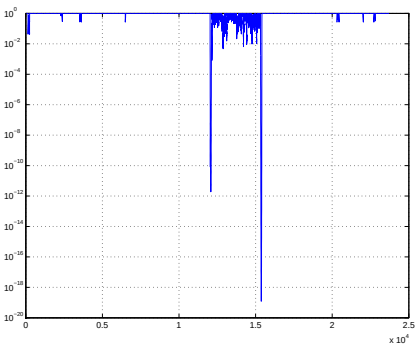

FIG. 5b : p-valeurs du test associé (IP3)

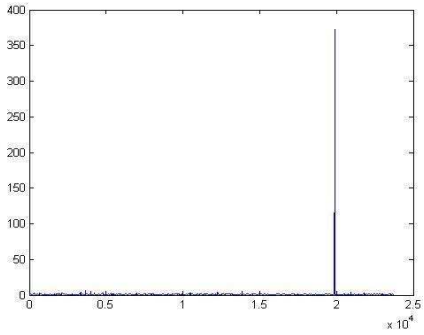

FIG. 6a : Série temporelle censurée de l'IP4

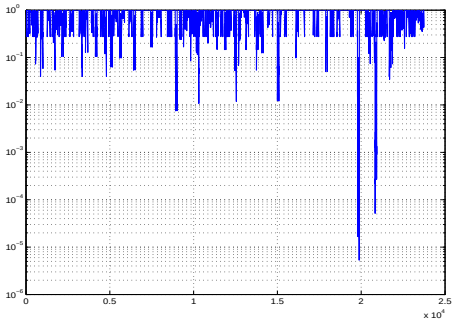

FIG. 6b : p-valeurs du test associé (IP4)

3.2 Performances numériques

Pour les valeurs des paramètres M=10, ntop=1 et *P*=100, le temps de calcul de l'algorithme présenté est largement satisfaisant puisque la durée totale du traitement est de 1 minute sur un PC muni d'un CPU de 3 GHz et d'une mémoire RAM de 1 Go, ce qui rend son implémentation en ligne tout à fait réaliste. Avec 0.0001 comme borne maximale pour les p-valeurs, notre algorithme appliqué à la trace décrite dans la section 2 (trace correspondant à 3 jours de trafic) détecte une vingtaine d'adresses IP potentiellement attaquées, les instants où se produisent ces attaques, les adresses IP source « attaquantes », les ports source et destination.

4. Discussion

On peut naturellement se demander si notre algorithme n'a pas tendance à détecter des attaques pour des adresses IP correspondant à des machines fortement sollicitées dans leur utilisation courante telles que les serveurs Web par exemple. De telles machines vont évidemment être sélectionnées dans l'étape du filtrage par records mais une détection d'attaque n'aura lieu que si un changement de régime effectif se produit dans l'évolution temporelle du nombre de paquets SYN reçus par cette machine (« Flash Crowd » par exemple).

D'autre part, la présence de telles machines ne nous empêche pas nécessairement de détecter des attaques moins intensives que pourraient subir d'autres adresses IP. Ce problème peut, en effet, être évité en augmentant ntop dans l'Étape 2, ce nombre devant être choisi en fonction du réseau auquel la méthode est appliquée et en particulier en fonction du nombre de serveurs présents.

Références

[1] D. Moore, G. Voelker and S. Savage, "Inferring Internet denial of service activity". *In Proc. of USENIX Security Symposium*, 2001.

[2] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, A. Lakhina. "Detection and identification of network anomalies using sketch subspaces". *Proceedings of the 6th ACM SIGCOMM on Internet measurement.* Pages: 147 – 152. Brazil, 2006.

[3] H. Wang, D. Zhang and K. G. Shin, "Detecting SYN flooding attacks", *in Proc. of IEEE INFOCOM'02*, 2002.

[4] A. Tartakovsky, B. Rozovskii, R. Blazek, and H. Kim, "Detection of Intrusion in Information Systems by Sequential Chang-Point Methods." *Statistical Methodology*, 2006: vol. 3, Issue 3, pp. 252-340.

[5] V. A. Siris **,** and F. Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks". *ICON 2004 - 12th IEEE International Conference on Network 2004.*

[6] E. Gombay, S. Liu, « A Nonparametric Test for Change in Randomly Censored Data ». *The Canadian Journal of Statistics*, Vol. 28, No. 1, pp. 113-121 (2000).