Compte-Rendu Cassiopée

Informations Générales :

Tuteur : Gregory Blanc
Membres du projet : Charles Mure, Félix Molina

Projet : **Détection d'intrusion par approche statistique sur un réseau SDN**

Github : https://github.com/CharlesMure/cassiope-NIDS

Repertoire des papiers de recherche : https://github.com/CharlesMure/cassiope-NIDS/tree/master/Docs

Semaine du 29/01/2018

1 – Analyse et extraction de données d'articles scientifiques concernant la détection d'intrusion

Analyse des articles :
(A1) **A Deep Learning Approch for NIDS** (Quamar Niyaz, Weiqing Sun, Ahmad Y Javaid, and Mansoor Alam, College Of Engineering, The University of Toledo)
(A2) **Deep learning approach for Network Intrusion Detection in Software Defined Networking** (Tuan A Tang ∗ , Lotfi Mhamdi ∗ , Des McLernon ∗ , Syed Ali Raza Zaidi ∗ and Mounir Ghogho∗†, ∗ School of Electronic and Electrical Engineering, The University of Leeds, Leeds, UK., †International University of Rabat, Morocco)

Le but est d'obtenir un NIDS à apprentissage supervisé (A2) ou non supervisé (A1), flexible et efficace (taux de détection élevé et faible taux de faux positif).

Attaques testées : DOS, Probe, U2R, R2L.

DataSet utilisés : NSL-KDD dataset – KDD Cup' 99. Les datasets sont non labelisés dans le cas de A1.

Nombres de parametres (features) :
Pour A1 : 121 avant classification ;
Pour A2 : 6 spécifiques (duration, protocole_type, src_bytes, dst_bytes, count, srv_count)

Différents types d'apprentissage : ANN (le plus rependu), SVM, NB, RF (Random Forest), SOM, OPF, CAPSNET.

Aglos de détection : STL (Self-Taught Learning utilisant le SMR) et SMR (Soft Max Regression).

Algos de classification : J48 Decision Tree Classifier, DMNB, SMR.

Etapes d'apprentissage :
Pour A1 :
1 – Unsupervised Feature Learning (UFL) réalisé avec l'algo de Sparse Auto-Encoder
2 – Classification (SMR)

Pour A2 :

Implémentation :
Pour A1 (dans le cas d'apprentissage STL):
1 – Discretisation des valeurs nominales des attributs (1-to-n encoding)
2 – Elimination de l'attribut ayant pour valeur 0
3 – Normalisation des valeurs entre 0 et 1 (max-min normalization)
4 – Sparse Auto-Encoder
5 – Soft Max Regression
Pour A2 :

Resultats :
Pour A1 : taux de détection → 98,84 % (avec STL) et 96,79 % (avec SMR)
Pour A2 : environ 70 % de taux de détection

2 – Pour la semaine du 05/02/2018

→ Récupération des datasets provenant du CIC (Canadian Institute for Cybersecurity)
    → Contacter a.habibi.l@unb.ca pour récupérer des Datasets réels

→ Développement de tests pour différents modèles de réseau de neurones (python Keras module)
→ Choix du modèle du système de réseau de neurones

→ Recherhcer les features accessibles sur OpenFlow

Semaine du 05/02/2018

1 – Sélection des premiers attributs à utiliser

→ IP source, IP destination
→ Port source, port destination
→ Protocole (type protocole)
→ Date, heure
→ Durée du flux
→ Nombre total de paquets forwardés
→ Nombre total de paquets backwardés
→ Taille moyenne des en-têtes forwardées
→ Taille moyenne des en-têtes backwardées
→ Débit

Semaine du 12/02/2018 et du 19/02/2018

1 – Création des premiers modèles de réseau de neurones

Dans un premier temps, nous avons créé quelques fonctions de preprocessing nous permettant, entres-autres, de binariser et normaliser les valeurs des attributs des datasets.

Le dataset que nous utilisons (KDD) contient 41 attributs et 6 classes (4 classes representant les types d'attaques, une classe dite « normal » et une classe dite « unknown »).

Puis, **nous avons tenté trois types de réseaux de neurones différents :**

1. Un **réseau de « base »** que nous avons conçu avec les exemples donnés dans la dcumentation de Keras (module Python utilisé). Il contient 7 layers suivant ce modèle :

Layer	Input	L1	L2	L3	L4	L5	L6	Output
Nb entrée	41	82	164	164	82	41	20	6

2. Un **modèle évolué basé sur la convolution mathématique**. Il provient d'un article de recherche (https://www.researchgate.net/publication/319717354_A_Few-shot_Deep_Learning_Approach_for_Improved_Intrusion_Detection by Md Moin Uddin Chowdhury∗, Frederick Hammond†, Glenn Konowicz‡, Jiang Li§, Chunsheng Xin and Hongyi Wuk, Department of Electrical and Computer Engineering, Old Dominion University)

TABLE I
CNN ARCHITECTURE FOR FEATURE EXTRACTION

Layer Index	Layer	Output Shape	Padding & Stride
1	Conv2D (64 filters, size: 1x3)	1, 38, 64	0,1
2	Conv2D (64 filters, size: 1x3)	1, 36, 64	0,1
3	Maxpooling2D	1, 18, 64	
4	Conv2D (128 filters, size: 1x3)	1, 16, 128	0,1
5	Conv2D (128 filters, size: 1x3)	1, 16, 128	1,1
6	Conv2D (128 filters, size: 1x3)	1, 16, 128	1,1
7	Maxpooling2D	1, 8, 128	
8	Conv2D (256 filters, size: 1x3)	1, 8, 256	1,1
9	Conv2D (256 filters, size: 1x3)	1, 8, 256	1,1
10	Conv2D (256 filters, size: 1x3)	1, 8, 256	1,1
11	Maxpooling2D	1, 4, 256	
12	Flatten	1024	
13	Fully Connected	100	
14	Dropout	100	
15	Fully Connected	20	
16	Softmax	5	

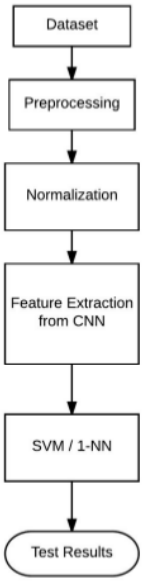

Fig. 1. Flowchart of our considered method.

3. **Un modèle CapsNet**, récupéré sur Github (https://github.com/XifengGuo/CapsNet-Keras).
A Keras implementation of CapsNet in the paper:
Sara Sabour, Nicholas Frosst, Geoffrey E Hinton. Dynamic Routing Between Capsules. NIPS 2017
(Non fonctionnel actuellement)

2 – Résultats obtenus et résultats attendus

1. Pour le premier modèle, le modèle de base, aucun résultat n'était attendu. Ce modèle nous a permis de nous familiariser avec le module Python Keras.

Notre résultat : Model Accurency : ~60 %

2. Pour le second modèle, nous attendions un resultat un peu inferieur à celui obtenu dans le papier de recherche (~95 % avec la fonction de classification) du fait que nous avions pas codé la fonction de classification.

Notre résultat : Model Accurency : ~75 %

3. Ce modele n'est pas encore fonctionnel mais selon les implémenatiosn déjà réalisées, les resultats sont de l'odre de 98 % avec une perte de 0,34 %.

3 – Puissance de calcul

Nos Pcs sont un peu limités pour une quantité de calcul aussi importante (environ un million multiplié par le nombre de d'entrenement (epoch)). Nous avons donc demandé un accès à un serveur de la platfome de TSP (octa-core cadencé server, 32Go de RAM).

Pour la suite...

1 - Etude détaillée des documentations des swiths SDN et réalisation de tests de récupération d'attributs via controlleur SDN.

2 – Validation du modèle lorsqu'on aura trouvé la configuration et la fonction de classification nous permettant d'avoir des resultats similaires à ceux obtenus par les papiers de recherches (voir mieux si possible).

3 – Revue pour Cassiopée (Gestion de projet) : Gantt, divison des tâches (sous-tâches), avancement avec reprise du cahier des charges.

4 -