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**

Le projet à pour objectif d'implémenter un système de détection complet basé sur des méthodes de DeepLearning et à partir des résultats de recherches issues de différents articles scientifiques parus en 2016 et 2017.
Le projet se focalise sur la mise en application des résultats de recherches dans un système capable de fonctionner en temps réel sur un réseau SDN. La partie DeepLearning du système sera mis en place en suivant les indications des différents articles de recherches et en utilisant des Framework permettant d'implémenter rapidement ces technologies (TensorFlow ou Caffe2 par exemple). Le projet se focalise sur la gestion des différentes entrées du système de détection et sur la mise en place d'algorithmes de traitement de flux de données permettant d'adapter le modèle de détection en continue. Ce système sera codé en Python.
La deuxième partie du projet consiste à la mise en place d'une interface web permettant d'interagir avec le système de détection, d'afficher les intrusions et de signaler les faux positif.
Pour finir, il est nécessaire d'obtenir un ensemble de données tests montrant l'activité
d'un réseau lors de différentes attaques afin de pouvoir réaliser la phase d'apprentissage initiale du système. Il existe actuellement diverses bases de données en ligne permettant de récupérer ces informations, mais elles sont pour la plupart ancienne. Avec l'aide de M.Blanc, nous espérons obtenir une base de données plus récente de la part de la communauté scientifique.

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

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

Structure du projet – Comparaison avec les objectifs initiaux

Le projet a été pensé comme un projet de recherche. Dans ce sens, la structure générale du projet initiale ainsi que les objectifs initiaux diffèrent un petit peu de la réalité.
Le projet est structuré en trois grandes parties, ne correspondant pas forcément à l'ordre de réalisation : Recherche, Production, Test. Les trois parties ne s'excluent pas. Nous avons fait beaucoup de recherches puis testé ce que nous trouvions sans obligatoirement y apporter une amélioration.

1 – Recherche

La partie de Recherche de notre projet est découpée en trois grands thèmes. Le premier étant les réseaux de neurones et les algorithmes utilisés, nous permettant de développer notre propre réseau. Le second thème regroupe toutes les recherches liées aux réseaux, virtuels ou non, aux SDN et aux protocoles d'extraction de données, nous permettant de construire notre réseau virtuel. Le troisième thème regroupe toutes les recherches annexes concernant l'implémentation de système de surveillance réseau fonctionnant en temps réel.

2 – Production

La partie Production concerne les deux sujets qui sont les piliers de notre projet : le réseau de neurones que nous devons produire, le plus performant possible, le plus facile et rapide à mettre en place et le réseau, physique ou virtuel, que nous devons mettre en place afin de tester en « condition réelle » le réseau de neurones et ainsi déterminer la productivité de notre système complet en vue d'une mise en production.

3 – Test

La partie de Test est une suite logique à la partie de Production. Ainsi, nous allons réaliser des batteries de tests afin de déterminer la meilleure implémentation du réseaux de neurones que nous pourrons produire. Puis nous testerons l'implémentation du réseau de neurones en condition quasi-réelle dans un réseau construit pour cette occasion.

En ce qui concerne les potentielles divergences entre le planning-projet initial et l'exécution réelle de ce planning, nous avons, pour l'instant, respecter la structure de notre projet à savoir Rechercher → Implémenter → Tester
et si besoin est Rechercher → Tester → Implémenter → Tester.

Cassiopée - NIDS 14 mars 2018

Tâches 2

Nom	Date de début	Date de fin
Analyse des papiers de recherche	29/01/18	06/02/18
Acquisition des jeux de données (datasets)	07/02/18	12/02/18
Mise en correlation des attributs provenant du SDN et ceux provenant du datasets	13/02/18	21/02/18
Choix des attributs utiles pour une implémentation SDN	13/02/18	21/02/18
Benchmark des différents modèles de réseaux de neurones	07/02/18	26/02/18
Implémentation de modèles	19/02/18	13/03/18
Implémentation Multi-Layer Perceptron	19/02/18	08/03/18
Implémentation Convolutional Model	19/02/18	13/03/18
Implémentation du Random Forest (Classifier)	05/03/18	13/03/18
Implémentation du SVM (Classifier)	05/03/18	13/03/18
Choix du modèle le plus performant	14/03/18	14/03/18
Test du modèle avec de nouveaux datasets	15/03/18	26/03/18
Correction de l'implémentation pour une "mise en production" sans apprentissage en continu	27/03/18	02/04/18
Développement des outils de collecte de données	03/04/18	09/04/18
Tests sans apprentissage en continu	03/04/18	15/05/18
Mise en place des outils de collecte des données	10/04/18	01/05/18
Liaison du programme et du controlleur SDN	03/04/18	01/05/18
Tests en condition réelle du programme	02/05/18	15/05/18
Amélioration du traitement des données acquises	16/05/18	22/05/18
Interface Web	16/05/18	29/05/18
Correction et finalisation pour une mise en production complète	16/05/18	04/06/18
Réalisation livrables	05/06/18	15/06/18

Les objectifs initiaux restent inchangés. Nous sommes actuellement dans la phase de « Test sans apprentissage continu », dans laquelle nous avons pris du retard à cause de problématique logicielle (Mininet et GNS3) et des collecteurs de données utilisés. Nous sommes donc légèrement en retard par rapport à notre planning initial. Ce retard peut-être rattrapé mais il y a un risque que notre prototype ne soit pas fonctionnel au 15/06/2018. (c.f. Risques). Nous avons quelques jours de marges grâce à la création de l'interface web qui existe déjà sur Github. En fonction de notre avancement, nous choisirons de la développer ou de l'installer prête à l'emploi moyennant quelques configurations de base.

Dans l'ensemble, nous avons respecté les échéances et les contraintes du projet. La seule tâche que nous ne réaliserons pas, par manque de temps et parce que c'est tout à fait inutile, est « le développement des outils de collecte de données » puisqu'ils existent déjà avec les fonctionnalités qui nous intéressent. Nous devrons cependant ajouter quelques lignes de code pour la récupération de certains attributs.

Analyse des risque – Solutions

Nous avons identifié dans notre projet 3 risques majeurs. Pour chaque risque identifié, nous proposons une solution alternative qui impactera la réalisation du prototype mais qui nous permettra cependant d'obtenir des résultats probant pour notre travail de recherche.

A la suite de nos recherches, nous avons identifié un protocole utilisé en entreprise pour surveiller les flux de données en entreprise: le protocole IPFIX. Ce protocole permet une collecte de flow à partir de différents point d'observation, de centraliser l'information et d'extraire plus de 400 attributs à partir de chaque flux (les attributs sont normés par l'IANA). Il s'agit d'un protocole dont nous avons peu de connaissance et qui risque d'être compliqué à utiliser correctement pour notre phase de tests. **Le risque est que nous n'arrivions pas à mettre en place correctement ce protocole sur un réseau SDN** pour réaliser une collecte des données en condition réelle. Si nous n'arrivons pas à mettre en place cette collecte, nous pourrons **utiliser des IDS existant** pour collecter les données et ensuite travailler à partir de leurs fichiers de logs générés mais il s'agit d'une solution plus lourde et moins personnalisable.

Le 2ème risque concerne la mise en relation des attributs des jeux de données d'apprentissage (contenant des exemples de flux d'attaques récent - 2015) et les attributs que nous pouvons collecter à l'aide du protocole IPFIX. En se basant sur une analyse statistique des jeux de données d'apprentissage, nous avons réduit le nombre d'attributs de 32 à 16 en perdant seulement 1% de précision dans notre classification. Dans ces 16 attributs, nous avons identifiés 5 attributs qui sont important pour la détection mais qui ne peuvent pas être extrait directement à l'aide du protocole IPFIX. Le **risque** est que nous n'arrivons pas à **implémenter la méthode de calcul de ces attributs à partir des attributs IPFIX.** Il s'agit ici d'un chemin critique car si nous n'arrivons pas à les re-générer, la solution serait de les écarter de notre modèle mais la précision de ce dernier serait alors grandement impacté.

Pour finir, il y a un **risque que la capacité de détection de notre modèle soit fortement corrélé à la typologie du réseau.** En d'autres termes, si la typologie du réseau sur lequel les données d'attaques ont été collectées est très différente du réseau de test en condition réelle, alors il y a un risque que notre prototype en phase de test obtienne des résultats peu probant. Cependant, nous avons essayé de réduire au maximum ce risque en sélectionnant des attributs ayant peu de rapport avec la typologie du réseau.
La solution possible si ce problème intervient est de mettre en place un réseau de test similaire à celui utilisé pour la collecte des données d'apprentissage (la typologie du réseau utilisé est documentée).

Compte-Rendu et Synthèse de l'avancement

Les tests réalisés sur les réseaux de neurones que nous avions pu construire (c.f. 18_Revue_1) nous ont permis de choisir un réseau afin de le soumettre à d'autres datasets et donc d'évaluer sa robustesse et sa fiabilité sur des jeux de données différents mais récents. Parallèlement, nous tentions de construire un réseau virtuel afin de simuler la récupération de trafic et l'envoi des flows vers un collecteur.

1 – Le réseau pour la production

Rappel Revue 1

Les tests réalisés juste avant la revue 1 nous ont permis d'obtenir les résultats suivants. Nous avons testé 4 types de réseaux avec calcul vectoriel du taux de détection :

→ Le réseau Multi Layer Perceptron ***seul (75%)***
→ Le réseau convolutionnel ***seul (77%)***
→ Le réseau Multi Layer Perceptron ***avec la classification SVM (76%)***
→ Le réseau convolutionnel seul ***avec la classification SVM (77%)***

Nous avons choisi de continuer à travailler seulement avec le réseau Convolutionnel.

2 – Tests du réseau de production

Afin de vérifier notre choix de réseau ainsi que notre implémentation que ce dernier, nous avons récupéré des jeux de données encore jamais utilisés depuis le début de notre projet. Il s'agit de **UNSW_NB15** disponible sur Github (https://github.com/FransHBotes/UNSW_NB15). Un papier de recherche en lien avec ces jeux de données a été publié, nous nous sommes basés dessus pour l'analyse des attributs utilisés, l'environnement d'obtention de ces trames.

Référence de l'article : Moustafa, Nour & Slay, Jill. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). 10.1109/MilCIS.2015.7348942.

L'avantage de ce dataset est que le papier de recherche associé est très explicite sur la sélection des attributs, nous n'avons donc pas besoin de sélectionner à nouveau. Les classes y sont aussi décrites. L'article est une justification en soi et le dataset est récent (2015).

Après quelques ajustements de l'implémentation du réseau de production (code pour production, simplification de l'algorithme de preprocessing des données), nous avons testé le réseau de production. Sans surprise, nous obtenons sensiblement les mêmes résultats qu'avec les précédents datasets. **Environ 78 % de détection** avec la correspondance vectorielle.

Par la suite, les attributs dont nous servons sont ceux présents dans le dataset.

Nous avons donc réussi à produire un réseau de neurones stable et robuste. Est-il le plus performant possible ? Nous avons du mal à en juger étant donné l'avancement de l'état de l'art à ce sujet. Mais nous pouvons considérer que notre implémentation est relativement correcte puisque les tests avec la correspondance absolue nous permettent d'obtenir 92 % de détection alors que les algorithmes utilisés dans les papiers de recherche obtiennent en moyenne 95 % de détection sans donner le taux de faux-positif ou même les optimisations effectuées.

3 – Construction du réseau virtuel de simulation

Le réseau de neurones étant prêt, nous devons maintenant réaliser un réseau de simulation. Par praticité, nous avons choisi d'utiliser un logiciel afin de simuler notre réseau. Ceci est plus simple au niveau logistique et au niveau des tests que nous pouvons faire sans avoir besoin d'avoir accès aux équipements physiques.
Pour cela, nous avons fait le choix d'utiliser le logiciel « Mininet » (http://mininet.org/) qui est un programme tournant sur linux, soit directement sur l'OS soit dans une machine virtuelle. Le logiciel est libre.
Nous avons aussi testé « GNS3 » (https://www.gns3.com/) qui a le même but que Mininet à savoir la virtualisation d'un réseau. Mais GNS3 propose beaucoup d'équipements CISCO qui ne sont pas libres et dont les images systèmes des équipements ne sont pas gratuites.

A ce jour, nous avons réussir à construire un réseau virtuel simple et suffisant pour faire nos premiers tests. Il se compose de deux terminaux linux connectés au switch SDN qui est lui même connecté à un réseau NAT permettant l'accès à internet.

Ces premiers tests constistent à profiter d'un protocole que les switchs SDN OVS (Open Virtual Switch) ont d'implémenter nativement dans leur système. Il s'agit du protocole IPFIX (IP Flow Information Export) qui, comme son nom l'indique, permet d'exporter les flows (ou trafic) IP vers un « Collecteur de flows IPFIX ».

Pourquoi ce protocole ? Son but est d'extraire des informations et il en extrait 482 exactement ce qui est un avantage de taille puisque ce sont 482 attributs potentiels qui peuvent être traités par le réseau de neurones. La deuxième raison de ce choix sont les calculs que le switch réalise. En effet, dans plusieurs datasets que nous avons testé, beaucoup avaient comme attributs des moyennes. Ces moyennes sont calculées par le switch sur demande du protocole IPFIX. IPFIX nous permet

donc de récupérer plus de 80 % des attributs dont nous avons besoin pour une bonne détection. Les 20 % restants seront récupérés directement sans passer par IPFIX.

A ce stade, il nous reste à choisir un collecteur. Un collecteur n'est rien d'autre qu'une sorte de boîte qui va recevoir les flows IPFIX dans un certain format et les afficher afin qu'elles soient lisibles par un humain. La différence entre les collecteurs du marché se situe sur la qualité des graphiques produits. Ils peuvent aussi collecter les informations d'autres protocoles d'export.
Le collecteur est un logiciel qui s'installe et lorsqu'il est lancé, va simplement écouter le trafic sur un port préalablement configuré.

Dans un premier temps, nous avons testé le collecteur « vFlow » (https://github.com/VerizonDigital/vflow/tree/v0.4.1). Nous n'avons encore pas réussi à faire fonctionner IPFIX entre notre réseau virtuel et le collecteur.
Nous avons une autre piste de recherche avec un autre collecteur que nous n'avons pas encore testé : SiLK (https://tools.netsa.cert.org/silk/index.html).

4 - Planification

Tâches

Nom	Date de début	Date de fin
Analyse des papiers de recherche	29/01/18	06/02/18
Acquisition des jeux de données (datasets)	07/02/18	12/02/18
Mise en correlation des attributs provenant du SDN et ceux provenant du datasets	13/02/18	21/02/18
Choix des attributs utiles pour une implémentation SDN	13/02/18	21/02/18
Benchmark des différents modèles de réseaux de neurones	07/02/18	26/02/18
Implémentation de modèles	19/02/18	13/03/18
Implémentation Multi-Layer Perceptron	19/02/18	08/03/18
Implémentation Convolutional Model	19/02/18	13/03/18
Implémentation du Random Forest (Classifier)	05/03/18	13/03/18
Implémentation du SVM (Classifier)	05/03/18	13/03/18
Choix du modèle le plus performant	14/03/18	14/03/18
Test du modèle avec de nouveaux datasets	15/03/18	26/03/18
Correction de l'implémentation pour une "mise en production" sans apprentissage en continu	27/03/18	02/04/18
Développement des outils de collecte de données	03/04/18	09/04/18
Tests sans apprentissage en continu	03/04/18	06/06/18
Mise en place des outils de collecte des données	10/04/18	15/05/18
Liaison du programme et du controlleur SDN	03/04/18	28/05/18
Tests en condition réelle du programme	29/05/18	06/06/18
Amélioration du traitement des données acquises	07/06/18	13/06/18
Interface Web	07/06/18	14/06/18
Correction et finalisation pour une mise en production complète	07/06/18	10/06/18
Réalisation livrables	11/06/18	21/06/18

Diagramme de Gantt

Nom	Date de d...	Date d...
Analyse des papiers de recherche	29/01/18	06/02/18
Acquisition des jeux de données (datasets)	07/02/18	12/02/18
Mise en correlation des attributs provenant du SDN et ceux prove...	13/02/18	21/02/18
Choix des attributs utiles pour une implémentation SDN	13/02/18	21/02/18
Benchmark des différents modèles de réseaux de neurones	07/02/18	26/02/18
⊟ Implémentation de modèles	19/02/18	13/03/18
Implémentation Multi-Layer Perceptron	19/02/18	08/03/18
Implémentation Convolutional Model	19/02/18	13/03/18
Implémentation du Random Forest (Classifier)	05/03/18	13/03/18
Implémentation du SVM (Classifier)	05/03/18	13/03/18
Choix du modèle le plus performant	14/03/18	14/03/18
Test du modèle avec de nouveaux datasets	15/03/18	26/03/18
Correction de l'implémentation pour une "mise en production" san...	27/03/18	02/04/18
Développement des outils de collecte de données	03/04/18	09/04/18
⊟ Tests sans apprentissage en continu	03/04/18	06/06/18
Mise en place des outils de collecte des données	10/04/18	15/05/18
Liaison du programme et du controlleur SDN	03/04/18	28/05/18
Tests en condition réelle du programme	29/05/18	06/06/18
Amélioration du traitement des données acquises	07/06/18	13/06/18
Interface Web	07/06/18	14/06/18
Correction et finalisation pour une mise en production complète	07/06/18	10/06/18
Réalisation livrables	11/06/18	21/06/18