VERS UNE NOUVELLE ARCHITECTURE DE DETECTION D'INTRUSION RESEAUX A BASE DE RESEAUX NEURONAUX

Berlin Hervé Djionang Lekagning, Gilbert Tindo

VERS UNE NOUVELLE ARCHITECTURE DE DETECTION D'INTRUSION RESEAUX A BASE DE RESEAUX NEURONAUX

Berlin H. LEKAGNING DJIONANG[1], *Gilbert* TINDO[2]
[1,2]Université de Yaoundé I, [1,2]Faculté des sciences, [1,2]Yaoundé, Cameroun
[1]dberlinherve@gmail.com, [2]gtindo@uycde.uninet.cm

RESUME : Les systèmes de détection d'intrusion réseau s'imposent aujourd'hui comme l'une des solutions efficaces pour détecter les activités illégales ou suspectes dans un réseau. Utiliser les réseaux de neurones pour pouvoir atteindre cet objectif a été étudié par plusieurs auteurs. La plupart des solutions proposées dans la littérature posent toujours le problème de pertinence et de fiabilité. L'une des causes majeures est la mauvaise conception du profil normal. Nous étudions dans ce papier l'importance de la sélection d'attributs dans le rendement des NIDS. Pour ce faire, nous proposons une architecture modulaire dans laquelle chaque module est construit pour détecter un type d'attaque et est entrainé d'abord avec tous les attributs, ensuite en élaguant certains attributs de l'ensemble d'apprentissage. Cette modularité nous permet de savoir quelle brique du système dégrade la performance du NIDS. L'expérimentation nous permet d'observer qu'en fonction du type d'attaque, certains attributs ont une influence marginale, et d'autres par contre ont une influence très prépondérante. Une étude comparative des performances de notre architecture a été effectuée. Le jeu de données NSL-KDD a été utilisé pour entrainer, tester et évaluer notre architecture.

ABSTRACT: Network IDS appear nowadays as one of most efficient solution to detect illegal/suspicions activities in network. Using neural networks to achieve this objective has been studied by many authors. Most of the solutions in the state of the art raise the pertinence and reliability problems. One of the major causes is the bad conception of the normal profile. In this paper, the importance of the selection of attributes in the efficiency on NIDS. To do it, we propose modular architecture in which every module is built to detect a type of attack and is train at first with all the attributes, then by pruning certain attributes of the whole learning. This modularity allows us to know which brick of the system degrades the performance of NIDS. The experiment allows us to observe that according to the type of attack, certain attributes have a marginal influence, and others on the other hand have a very dominating influence. A comparative study of the performances of our architecture was made. The NSL-KDD dataset has been use for training, testing and evaluating our architectures.

MOTS-CLES : système de détection d'intrusion réseaux, réseaux de neurones, MLP, KDD Dataset
KEYWORDS: NIDS, neural network, MLP, KDD Data set

1. INTRODUCTION

L'avènement des réseaux offre les services immenses à ceux qui l'utilisent. Ces services font l'objet de plusieurs attaques et les mécanismes de sécurité deviennent une nécessité pour les protéger. Les systèmes de détection d'intrusion réseaux (NIDS) sont l'un des mécanismes le plus utilisé aujourd'hui pour détecter les intrusions. Le but des systèmes de détection d'intrusion réseaux est de protéger les réseaux des attaques qui ne peuvent pas être identifiés par des firewalls.

Les systèmes de détection d'intrusion sont généralement classés en trois catégories [7] : les IDS qui utilisent l'approche comportementale (ceux qui cherchent à détecter les anomalies), les IDS qui utilisent l'approche par scénarios (ceux qui cherchent à détecter les signatures) et les IDS qui utilisent l'approche par spécification. L'analyse comportementale a généralement deux phases : une phase d'apprentissage et une phase de détection. Au cours de la phase d'apprentissage, le système apprend à reconnaitre ce qu'est un comportement normal, et au cours de la phase de détection, il identifie les comportements anormaux [6]. L'analyse des scenarios n'a pas de phase d'apprentissage. Ils viennent avec une base de scénarios d'attaques prédéfinis qu'ils doivent reconnaitre (signature). L'approche par spécification est une méthode qui construit le profil normal sans utiliser les algorithmes d'apprentissage [8].

Plusieurs méthodes d'apprentissage de profils (statistique, apprentissage artificiel et système expert) de référence ont été utilisées dans la littérature. Les réseaux de neurones ont été utilisés par plusieurs auteurs pour construire un système de détection d'intrusion réseaux [11, 12, 13, 15, 16, 17]. L'un des problèmes majeur des NIDS est que la performance est régie par un seul grand système qui se charge de détecter soit les types, soit les catégories d'attaques. Nous proposons une architecture modulaire de détection d'intrusion réseau par type d'attaque à l'aide des réseaux de neurones. Ceci nous permettra de savoir quel module tire les performances de l'ensemble vers le bas. Notre NIDS est basé sur l'approche comportementale.

L'un des problèmes majeurs de cette approche est la conception du profil normal [9]. La génération automatique des champs qui compose le profil d'un utilisateur est un véritable défi. La qualité de ces champs est un facteur important dans l'efficacité de l'IDS [10]. Nous expérimenterons dans ce travail, l'importance de la sélection d'attribut en vue de la construction de bons profils.

La suite de notre papier est organisée comme suit : nous allons présenter et justifier le choix du modèle de réseaux de neurones pour l'apprentissage des profils de référence à la section 2 ; à la section 3 nous avons fait une bref revue des travaux liés aux systèmes de détection d'intrusion à base de réseaux de neurones ; à la section 4, nous proposons notre architecture et son fonctionnement ; la section 5 est consacrée à la description des données d'entrée et le prétraitement effectué sur ces dernières ; nous passons à l'expérimentation et l'analyse des résultats de nos travaux aux sections 6 et 7 ; nous terminons ce travail avec une conclusion à la section 8.

2. RESEAUX DE NEURONES

Les réseaux de neurones sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle et reliés par des poids. Ces poids de connexion gouvernent le fonctionnement du réseau. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Les réseaux de neurones ont plusieurs avantages dans la mise en œuvre d'un système de détection d'intrusion. Ils sont très efficaces et rapides dans la tâche de classification [11]. Ils sont capables d'apprendre et d'identifier facilement les nouvelles menaces qui leur sont soumises. Les réseaux de neurones sont capables de traiter les données incomplètes, imprécises et provenant de sources multiples. La rapidité naturelle des réseaux de neurones permet de réduire les dommages lorsque la menace est détectée [12]. L'utilisation des réseaux de neurones permet d'extraire les relations non linéaires qui existent entre les différents champs d'un paquet et de détecter en temps réel les attaques complexes [13]. Les réseaux de neurones, après avoir appris correctement, ont une bonne capacité de généralisation, c'est-à-dire qu'ils sont capables de calculer avec précision les sorties correspondantes même pour des données qui n'ont pas été apprises. La flexibilité qu'offrent les réseaux de neurones est également l'un des atouts pour la détection d'intrusion [14].

3. TRAVAUX ANTERIEURS

James Canady [12] est le premier chercheur à proposer un système de détection d'intrusion basé sur l'analyse des protocoles réseau à base de réseaux de neurones. Il utilise les perceptrons multicouches (MLP) pour l'apprentissage. Le tableau ci-dessous (**Tableau 1**) présente quelques résultats des chercheurs ayant utilisés le modèle de réseaux de neurones pour concevoir un IDS réseau. Le processus d'apprentissage utilisé pour modifier les paramètres du réseau utilisé par ces chercheurs sont [20] :

- ✓ Présenter au réseau de neurones les données sous forme d'un vecteur
- ✓ Vérifier si la sortie générée correspond à la sortie désirée
- ✓ Changer les paramètres du réseau afin de tendre vers la sortie désirée.

Ces travaux utilisent le jeu de données KDD pour valider leur modèle.

Critères / Auteur	Taux de reconnaissance	Faux Positif	Faux négatif
Alan 2002 [19]	24%	-	-
Mehdi 2004 [20]	87%	-	-
Golovko 2005 [13]	94,3%	-	-
Vaitsekhovich 2008[15]	93,21%	12,90%	-
Khattab 2009 [18]	97%	2,4%	0,8%
Iftikar 2009 [21]	98%	1,5%	-
Muna 2010 [17]	78%	-	-
Aslihan 2012 [16]	93,42%	2,95%	-
Yousef 2012 [14]	95,4%	2,6%	-

Tableau 1 : Résultats de quelques travaux

Iftikar 2009 [21] détecte uniquement les attaques de types scan.

ARIMA

4. ARCHITECTURE PROPOSEE ET FONCTIONNEMENT

Nous proposons dans cette section notre architecture, par la suite nous décrivons le fonctionnement de cette dernière et nous dégageons et démontrons une propriété de notre architecture.

4.1 Architecture

Nous proposons une architecture modulaire dans laquelle chaque module permet de détecter uniquement un type d'attaque. Elles discriminent chaque type d'attaque aux paquets normaux. La modularité de notre architecture permet de régler la performance du NIDS. Nous proposons des modèles bi-neuronaux pour cela. Nous définissons les réseaux bi-neuronaux comme des réseaux qui discriminent deux classes. Notre architecture (Figure 1) propose un modèle de discrimination binaire (B_MLP) sans sélection d'attributs et un modèle de discrimination binaire (SB_MLP) avec sélection statique d'attributs. L'objectif du deuxième modèle étant de voir l'impact de la sélection des attributs sur la décision du classifieur. Pour m types d'attaques, nous devons disposer de $m+1$ modèles (B_MLP) pour notre architecture. Notre architecture a quatre niveaux dont le premier d'effectuer les prétraitements sur les données. Le deuxième permet de discriminer les paquets normaux des paquets anormaux. Si le paquet analysé est anormal alors le paquet est passé aux autres modèles (troisième niveaux) afin de déterminer le type d'attaque. L'élément **A** (quatrième niveau) dans ces architectures désigne un arbitre qui décidera de quel type d'attaque il s'agit. L'algorithme qui permet d'apprendre ces réseaux est décrit à la **figure 2**. Chaque modèle est constitué de trois couches dont une couche d'entrée, une couche cachée et une couche de sortie. Pour chaque modèle, nous recherchons le nombre de neurones de la couche cachée qui permet d'obtenir le meilleur taux de détection dynamiquement.

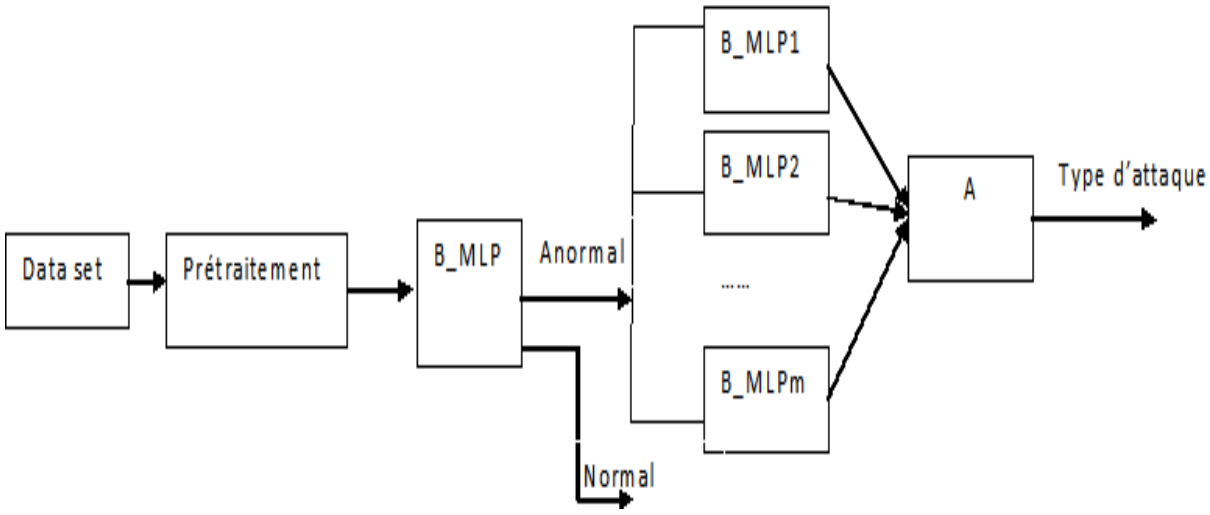

Figure 1 : Architecture de détection d'intrusion à 4 niveaux

4.2 Fonctionnement

Pour chaque vecteur de données, on effectue un prétraitement comme mentionné dans la section 5. L'algorithme de rétropropagation des erreurs est utilisé pour l'apprentissage de chacun de nos modèles binaires neuronaux. Chaque modèle bi-neuronaux doit être optimal dans la classification de chaque type d'attaque de telle sorte que le taux de reconnaissance se rapproche de 100.

Figure 2 : **Algorithme exécuté par chaque réseau**

1. *Normaliser les données d'entrées.*
2. *Rechercher pour chaque réseau le nombre de neurones de la couche cachée qui donne le meilleur taux de reconnaissance et stocker ce nombre dans un fichier.*
3. *Apprendre chaque réseau à l'aide de l'algorithme de rétropropagation. Et stocker les poids obtenus dans un fichier.*
4. *Evaluer nos réseaux avec les données d'évaluation. Ceci est effectué à l'aide des poids stocké lors de l'apprentissage et le nombre de neurones de la couche cachée obtenues.*

Phase d'entrainement et test

Il faut entrainer B_MLP avec l'ensemble de données d'entrainement et le tester avec les données test. Faire de même pour B_MLPi par l'ensemble d'entrainement correspondant aux paquets normaux et leurs types respectifs.

Phase d'évaluation

Le premier niveau de notre réseau discrimine les profils normaux des profils anormaux. Si le vecteur de donnée lu appartient à la classe des paquets anormaux, nous le passons aux autres modèles binaires neuronaux. Chaque modèle produit une sortie et l'arbitre décidera en fonction de ces sorties de quel type d'attaque il s'agit. L'arbitre se contentera de gérer un vecteur de m sorties. La sortie la plus petite détermine le type d'attaque. Et si toutes les sorties sont supérieures à la valeur de tolérance ou critère de classification alors il s'agit d'une attaque de type inconnue.

4.3 Propriété

Soit m le nombre de type d'attaques à détecter, si n est la taille des données d'entrée et p le nombre de neurones de la couche cachée de chaque modèle bi-neuronaux (B_MLP). La taille de l'architecture en terme de nombre de neurones est de l'ordre de O ($p * n * m$).

Preuve : La taille de notre réseau est de $(p * n + 1)(m + 1)$.

5. DESCRIPTION DES DONNEES ET PRETRAITEMENT

Depuis 1999, KDD Cup 99 est utilisé comme ensemble de données exemple dans les systèmes de détection d'intrusion comportementale [4]. Chaque paquet de l'ensemble des données de KDD Cup 99 est constitué de 41 champs et est labellisés comme paquet normal ou paquet anormal avec les types d'attaques. Parmi ces champs, 37 sont des champs de type numérique et 4 sont des champs de type non numérique. KDD99 regroupe 37 types d'attaque. Ces attaques sont divisées en quatre grandes classes : DOS, U2R, R2L et Probes [1,2].

- **DOS (Denial of service attaks):** ce sont les attaques qui visent à porter atteinte à la disponibilité des services en saturant les ressources de la machine, serveur ou réseau cibles. Ces attaques réussies dans les réseaux ont pour conséquence immédiate le blocage du trafic réseaux.
- **Probes :** attaque qui vise à réunir les informations sur la cible susceptible d'aider l'attaquant à initier une attaque. Il existe plusieurs types d'attaques probes: certains abusent les utilisateurs légitime et d'autres utilisent la technique d'ingénierie pour collecter les informations.
- **R2L (Remote To Local) :** attaque qui vise à contourner ou usurper les paramètres d'authentification d'une cible afin d'exécuter les commandes. La plupart de ces attaques sont issues de la sociale ingénierie [1].
- **U2R (User To Root) :** l'attaque provient ici de l'intérieur. L'attaquant usurpe le mot de passe du super administrateur par conséquent des autres utilisateurs. La plupart de ces attaques sont issues de la saturation du Buffer causée par les erreurs de programmation [1].

Les données KDD99 regorgent beaucoup de paquets redondants dans les données d'entrainement comme de test [4]. Les données redondantes sont capable de donner plus d'importance à un type d'attaque qu'elle ne le mérite. [4] propose NSL-KDD qui est un jeu de données excellent pour comparer les IDS réseaux. Notre expérimentation est effectuée avec NSL-KDD, le type d'attaque et leurs nombres dans le jeu de données d'entrainement et de test sont proposés dans le **tableau 2** en appendice.

5.1 Prétraitement

Le prétraitement porte sur les champs non numériques. Les champs non numériques sont : le type de protocole (TCP, UDP, ICMP), le type de service (aol, auth, bgp, … Z39_50), le drapeau (OTH, REJ, RSTO, RSTOS0, RSTR, S0, S1, S2, S3, SF, SH) et la classe du paquet (Normal ou Anormal). Pour le type de protocole, nous affectons les valeurs numériques suivantes : TCP=1, UDP=2 et ICMP=3. Nous affectons 1 aux paquets normaux et 0 aux paquets anormaux. Pour les champs de types service et drapeau, nous pouvons leur attribuer les valeurs numériques par ordre croissant ou décroissant de leur nombre total. [5] a montré les limites d'une telle approche, il propose de donner les valeurs aléatoires à ces champs. Nous avons dans nos travaux donnés les valeurs aléatoires entre 1 et 10 aux champs de type drapeaux et les valeurs aléatoires entre 1 et 65 aux champs de type services.

5.2 Normalisation

Il s'agit de transformer les données pour qu'elles varient entre 0 et 1 afin de rendre les données plus homogènes et ainsi simplifier l'apprentissage des réseaux. Nous allons dans ce papier utiliser la normalisation Min-Max. Soit min_xet man_x le minimum et le maximum respectifs des valeurs de l'attribut X de valeur V, la valeur normalisée $\boldsymbol{V'=\frac{v-min_x}{man_x-min_x}}$. Pour chaque attribut du vecteur de données, calculer sa valeur normalisée et remplacer cette dernière par la valeur normalisée.

6. EXPERIMENTATIONS ET RESULTATS

L'expérimentation est effectuée sur nos deux modèles. Les résultats par types d'attaques et par catégories d'attaques sont représentés sur la **Figure 5** et dans le **tableau 3** respectivement. Pour évaluer nos modèles, nous allons utiliser quatre indicateurs qui sont : le taux de reconnaissance (TR), le taux de faux positif (TFP), le taux de détecté (TR) et le taux de faux négatif (TFN). Ces taux sont évalués comme s**uit :** $\boldsymbol{TR=\frac{NN+AA}{NN+AA+AN+NA}*100}$,$\boldsymbol{TFP=\frac{NA}{NA+AA}*100}$,$\boldsymbol{TFN=\frac{AN}{AN+NN}*100}$, $\boldsymbol{TD=\frac{AA}{AA+NA}*100}$ avec :

NN : paquet Normal détecté comme Normal ; **NA** : paquet Normal détecté comme Anormal ; **AN** :paquet Anormal détecté comme Normal ; **AA** :paquet Anormal détecté comme Anormal.

Pour l'expérimentation, 80% des données sont utilisées pour l'entrainement dans lequel 20% sont réservées pour l'évaluation et 20% des données sont utilisées pour le test. L'ensemble des données que nous soumettons à chaque réseau est réduit par rapport aux données initiales.

Les résultats expérimentaux sont représentés dans les tableaux (3 et 4) et les courbes comparatives (3, 4, 5 et 6). **TA** représente le type d'attaque sur les courbes présentées.

Nous disposons de deux modèles de notre architecture, la première (Arch_m1) qui ne prend pas en compte la sélection d'attribut et la deuxième(Arch_m2) qui prend en compte.

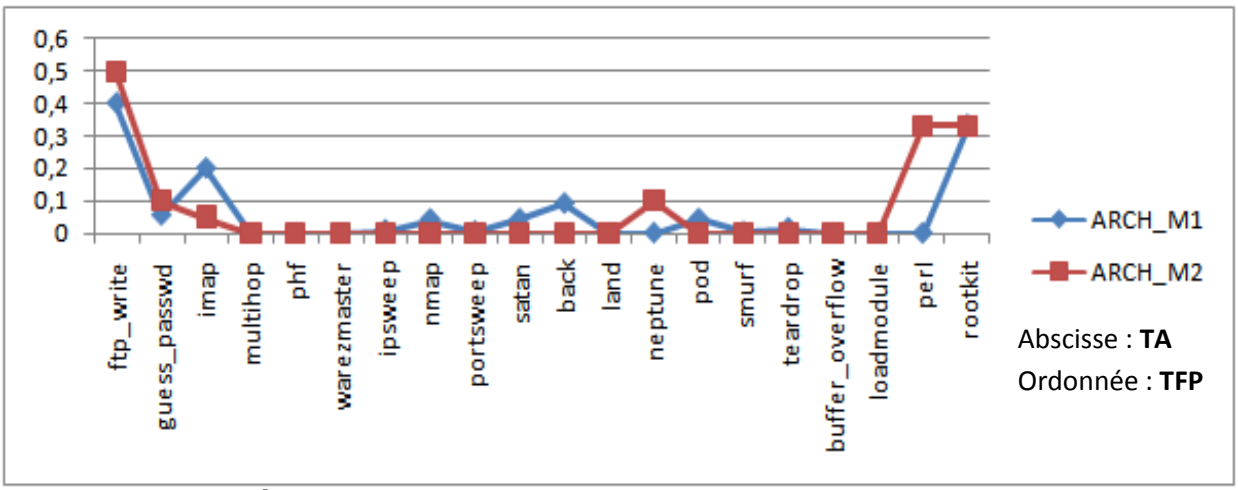

Figure 3 : Étude comparative des taux de faux positifs des deux modèles

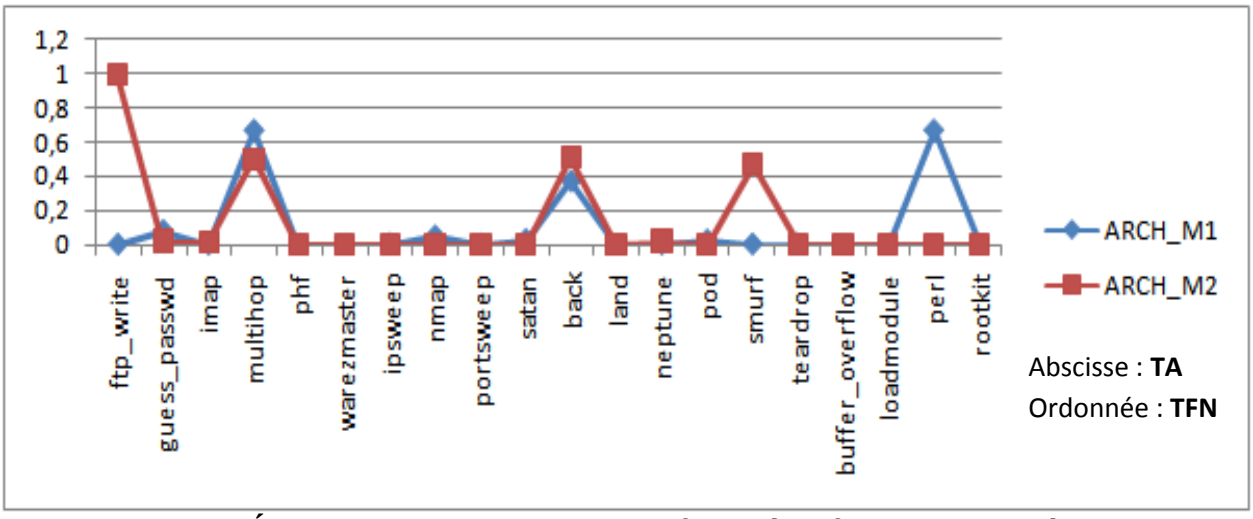

Figure 4 : Étude comparative des taux de faux négatifs des deux modèles

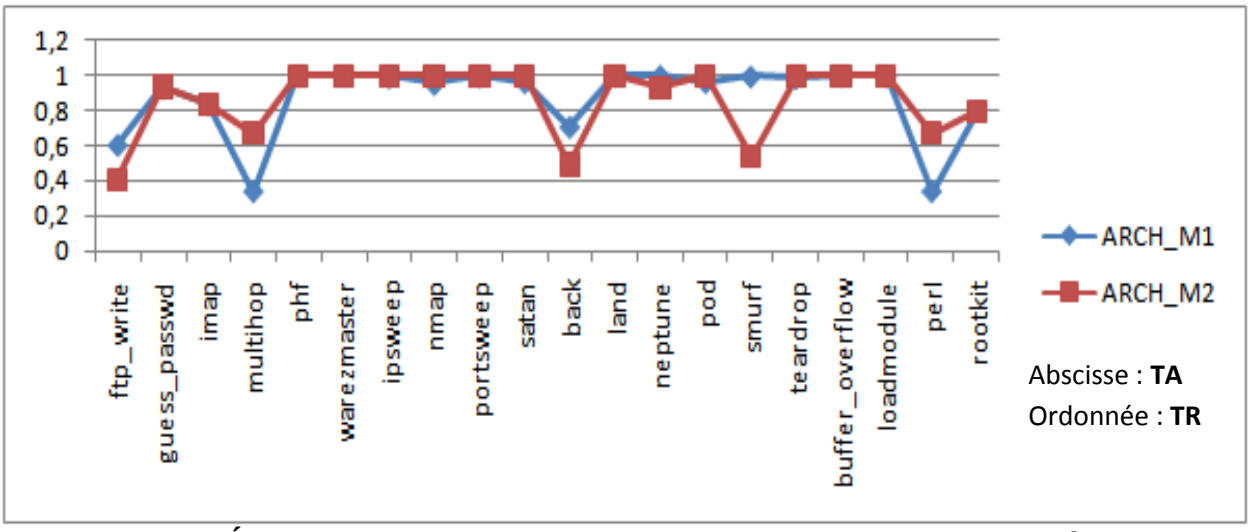

Figure 5 : **Étude comparative des taux de reconnaissances des deux modèles**

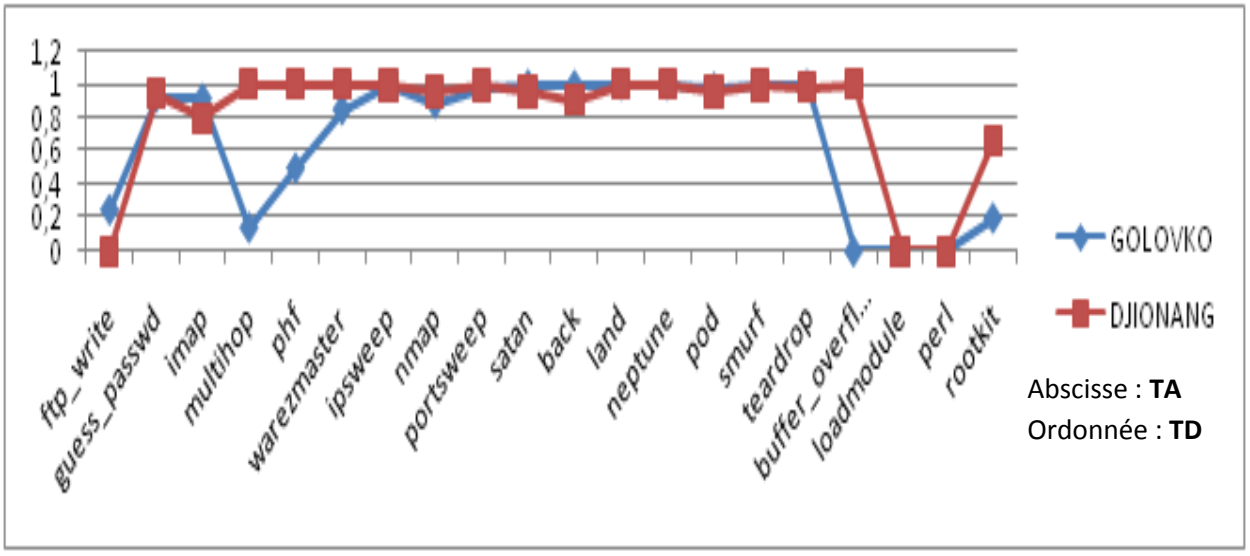

Figure 6 : **Étude comparative avec les travaux de Golovko**

ARIMA

7. ANALYSE DES RESULTATS

Nous avons pour notre deuxième architecture fixée 12 attributs qui sont : durée, protocole, service, drapeaux, adresse source, adresse destination, land, wrond_fragment, urgent, hot et la classe (Normal, Anormal). Au regard des résultats présentés dans le **tableau 2**, on remarque que pour certains types d'attaques, le taux de reconnaissance diminue. Ceci est la preuve que cet ensemble d'attributs n'est pas pertinent pour détecter les attaques de ce type. Un besoin important s'impose au niveau du choix des attributs pertinents qui peuvent représenter chaque attaque.

Les figures (3 et 4) présentent une étude comparative des deux modèles par rapport au taux de faux positif (**Figure 3**) et taux de faux négatif (**Figure 4**). Les pics (ftp_write, back, smurf) que nous observons sur la figure 4 nous enseignent sur le fait que la mauvaise sélection des attributs pertinents peut augmenter le taux de faux négatif ou positif.

Le **tableau 4** nous permet de comparer nos résultats avec les résultats de [13]. Le taux de détecté est utilisé parce que c'est la mesure qu'utilise Golovko pour présenter ces résultats. Nous l'avons choisi pour comparer nos résultats tout simplement parce qu'il détecte les types d'attaque comme notre architecture. La **Figure 6** montre bien comment la courbe représentant nos résultats est nettement au-dessus de la courbe représentant les travaux de Golovko. Nos résultats comparés aux travaux de Golovko sont généralement meilleurs et plus observables particulièrement pour les attaques du type R2L.

Catégorie	Fréquence	Taux de Reconnaissance	
		Arch_m1	Arch_m2
R2L	103	86%	71,8%
PROBES	12630	96%	96,12%
DOS	44953	99,9%	91,51%
U2R	52	92,3%	94,2%

Tableau3 : Etude comparative des catégories d'attaque

Le **tableau 3** nous permet de voir clairement comment la sélection des attributs non représentatifs peut influencer le taux de reconnaissance global. Le cas des attaques du type DOS est plus visible. Nous partons de 99,9% de taux de classification dans **l'architecture modèle 1** tandis que les résultats de **l'architecture modèle 2** ramènent le taux de reconnaissance à 91,51%. Ceci est révélateur d'un fait : le choix des données d'entrée ont un impact considérable sur la qualité d'un système de détection d'intrusion.

La **figure 5** nous montre que globalement, la courbe représentant les résultats de **l'architecture modèle 2** est presque toujours au-dessus de la courbe représentant les résultats de **l'architecture modèle 1.** Ceci est la preuve que la sélection lorsqu'elle est bien menée permet d'améliorer la qualité des IDS.

Type d'attaque	Taux de Détecté (TD)	
	GOLOVKO 2005	DJIONANG 2015
ftp_write	25%	-
guess_passwd	92,45%	94,44%
imap	91,67%	80%
multihop	14,29%	100%
phf	50%	100%
warezmaster	85%	100%
ipsweep	99,12%	99,06%
nmap	88,74%	95,74%
portsweep	98,81%	99,44%
satan	99,81%	95,35%
back	99,5%	90,83%
land	100%	100%
neptune	99,98%	99,93%
pod	98,11%	95,53%
smurf	100%	99,63%
teardrop	99,8%	98,34%
buffer_overflow	0%	100%
loadmodule	0%	-
perl	0%	-
rootkit	20%	66,67%

Tableau 4: Etude comparative

8. CONCLUSION

Nous avons dans cet essai, proposé une architecture modulaire des systèmes de détection d'intrusion réseau à l'aide des réseaux neuronaux. La modularité de nos architectures permet d'enlever les briques qui portent atteinte à la performance de l'IDS. Notre travail nous permet de montrer l'intérêt de la sélection d'attribut. Nos expérimentations ont montré l'intérêt de réduire la dimension des échantillons d'apprentissage et de test. L'architecture avec une couche de sélection d'attribut permet d'obtenir une bonne classification pour certains types d'attaques. Comment donc caractériser les données d'entrée afin de construire un bon profil de paquets normaux? Comment identifier les champs qui permettent de caractériser les différentes attaques ?

1. REFERENCE

[1] Srinivas Mukkamala && all "Intrusion detection using an ensemble of intelligent paradigms", Journal Network and Computer Applications 28 (2005), 167-182

[2] Matthew Vincent Mahoney "A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic ", these of Florida Institute of Technology, May 2003

[3] Mohammad Khubeb Siddiqui and Shams Naahid "Analysis of KDD CUP 99 Dataset using Clustering based Data Mining", International Journal of Database Theory and Application Vol.6, No.5 (2013), pp.23- 34

[4] Mahbod Tavallaee && all "A Detailed Analysis of the KDD CUP 99 Data Set" Proceding of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Application (CISDA 2009)

[5] Aslihan Ozkaya && Bekir Karlik "Protocole Type Based Intrusion Detection Using RBF Neural Network" International Journal of Artificial Intelligence and Expert Systems(IJAE), volume(3): Issue(4):2012

[6]: Ludovic Mé and Véronique Alanou. Détection d'intrusion dans un système informatique : méthodes et outils. TSI, 15(4):429–450, 1996.

[7] Asmaa Shaker, Sharer Gore « Importance of Intrusion Detection System » International Journal of Scientific & Engineering Research, Janvier 2011.

[8] Guillaume Hiet, « Détection d'instructions paramétrée par la politique de sécurité grâce au contrôle collaboratif des flux d'informations au sein du système d'exploitation et des applications : mise en œuvre sous linux pour les programmes java » Université de Rennes, Decembre 2008

[9] Ning P. && Jajodia S. Intrusion Detection Techniques. In H. Bidgoli (Ed.), The Internet Encyclopedia. John Wiley & Sons. (2003).

[10] Guofei Gu & al "Towards an Information-Theoretic Framework for Analyzing Intrusion Detection Systems" Computer Security ESORICS 2006 (2006): 527-546

[11] G. DREYFUS "les réseaux de neurones" Mécanique Industriel et Matériaux, n51, septembre 1998

[12] Cannady,J. , "Artificial Neural Network s for Misuse Detection," Proc eedings, National Information Systems Security Conference (NISSC '98), October, Arlington , VA, pp . 443 -456. 1 998

[13] Vladimir Golovko, Pavel Kochurko "Intruision recognition using neural networks" International Scientific Journal of computing, 2005, vol. 4, Issue3, 37-42

[14] Yousef Abuadlla && all "Flow-Based Anomaly Intrusion Detection System Using Two Neural Network Stage", Computer Science and Information systems 11(2): 601-622: 2012

[15] Leanid VAITSEKHOVICH, Vladimir GOLOVKO « Employment of neural network baser classifier for intrusion detection » acta mechanica et automatica, vol2, no4 ,2008

[16] Aslihan Ozkaya && Bekir Karlik "Protocole Type Based Intrusion Detection Using RBF Neural Network" International Journal of Artificial Intelligence and Expert Systems(IJAE), volume(3): Issue(4):2012

[17] Muna Mhammad && Monica Mehrotra "Design Network Intrusion Detection System using Hybrid Fuzzy-Neural Network" International Journal of Computer Science and Security, volume(4): Issue(3): 2010

[18] M. Khattab Ali && all "The Affect of Fuzzification on Neural NetworksnIntrusion Detection System", IEEE computer society: 2009

[19] ALAN BIVENS && all "Network based intrusion detection using neural network" Intelligent Engineering Systems through Artificial Neural network, 2002, pp. 579-584

[20] Mehdi MORADI and Mohammad ZULKERNINE, "A Neural Network based System for intrusion detection and Classification of Attacks" In 2004 IEEE International on Advances in Intelligent Systems.

[21] Iftikhar Ahmad && all "Application of Artificial Neural Network in Detection of Probing Attacks" 2009 IEEE Symposium on Industrial Electronics and Applications(ISIEA 2009), October 4-6, 2009, Kuala Lumpur, Malaysia

APPENTICE

Les attaques par catégories du jeu de données NLS-KDD99. Ces données sont organisées en terme de données d'entrainement et de test.

Catégorie	Type d'attaque	Entrainement	Test	Catégorie	Type d'attaque	Entrainement	Test
Normal	Normal	67 343	9711	DOS	neptune	41214	4657
R2L	ftp_write	8	3		pod	201	41
	guess_passwd	53	1231		processtable	0	685
	httptunnel	0	133		smurf	2646	665
	imap	11	1		teardrop	892	12
	multihop	7	18		udpstorm	0	2
	named	0	17	U2R	buffer_overflow	30	20
	phf	4	2		loadmodule	9	2
	sendmail	0	14		perl	3	2
	snmpgetattack	0	178		ps	0	15
	snmpguess	0	331		rootkit	10	13
	warezmaster	20	944		sqlattack	0	2
	worm	0	2		xterm	0	13
	xlock	0	9				
	xsnoop	0	4				
Probes	ipsweep	3599	141				
	mscan	0	996				
	nmap	1493	13				
	portsweep	2931	157				
	saint	0	319				
	satan	3633	735				
DOS	apache2	0	734				
	back	956	359				
	land	18	7				
	mailbomb	0	293				

Tableau 2 : Type d'attaque