L'impact du choix des données d'apprentissage dans la génération des IDS par la programmation génétique

Sana Ben Hamida, Rabiaa Djebali, Khaled Ghedira

L'impact du choix des données d'apprentissage dans la génération des IDS par la programmation génétique

Rabiaa DJEBALI*, Sana Ben HAMIDA, Khaled GHEDIRA*****

* Laboratoire (LI3)- École Nationale Des Sciences de l'Informatique–Université Mannouba- Tunis
Rabiaa.djebali@yahoo.fr
** Laboratoire (LI3)- École Supérieure de Technologies Et d'Informatique
Bhsana@yahoo.fr
*** Laboratoire (LI3)- École Nationale Des Sciences de l'Informatique–Université Mannouba- Tunis
khaled.ghedira@isg.rnu.tn

RÉSUMÉ. *Dans cet article, on présente une méthodologie de génération automatique d'un modèle décisionnel pour la classification des connections Internet basée sur la programmation génétique. Il introduit aussi une étude comparative entre deux techniques de sélection des échantillons des données d'apprentissage au cours de l'évolution et leurs impacts sur la qualité des modèles de classification obtenus.*

ABSTRACT. *In this article, one presents an automatic methodology of generation of a decisional model for the classification of connections Internet based on the genetic programming. It introduces also a comparative study between two techniques of selection of the samples of the data for training during the evolution and their impacts on the quality of the models of classification obtained.*

MOTS-CLÉS : *sécurité, intrusion, IDS, programmation génétique, apprentissage, KDD.*

KEYWORDS: *security, intrusion, IDS, genetic programming, training, KDD.*

1 Introduction

Les systèmes d'information sont aujourd'hui de plus en plus ouverts sur Internet. Cette ouverture présente un moyen révolutionnaire pour la communication et l'échange de l'information et présente aussi un domaine pour l'intrusion et l'attaque des données. D'où la nécessité de mise en place d'un système de détection d'intrusion (IDS: « Intrusion Detection Systems ») pour auditer le système d'information et détecter d'éventuelles intrusions.

La mise au point des IDS (S. Mukkamal et al, 2004) s'affronte à plusieurs défis dont l'adaptation au changement permanent de l'environnement et le changement des critères d'un comportement normal et anormale. De plus, un IDS doit fonctionner avec un minimum d'erreurs (le taux des fausses alarmes doit être négligeable).

De ce fait, plusieurs activités de recherche dans le monde s'orientent actuellement vers la mise au point des nouvelles techniques pour améliorer la performance des IDS comme les réseaux de neurones, les systèmes multi agents, les algorithmes évolutionnaires et datamining [P.Biondi, 2001].

Dans notre travail, nous allons commencer par présenter l'approche évolutionnaire choisie (programmation génétique) pour résoudre le problème de détection d'intrusions. Par la suite, dans la section 2, nous présentons une description de la base d'apprentissage KDD (*Kristpher Kandall Database*). Ensuite, dans la partie expérimentation, nous présentons la plateforme sur laquelle nous avons travaillé et les paramètres du programme génétique. Enfin, nous présentons les différents résultats obtenus avec une étude comparative.

2 L'approche GP pour la détection d'intrusion

Pour encourager les recherches dans le domaine de la détection des intrusions Internet, un challenge a été lancé au cours de la 5ème SIGKDD (5th International Conference on Knowledge Discovery & Data Mining 1999). 24 propositions ont été soumises par des différentes équipes de recherche dans le monde. Les trois solutions gagnantes sont basées sur les arbres de décision et elles sont très coûteuses. Par exemple, la deuxième solution est un « forêt » de 755 arbres (I. Levin, 2000) et la troisième nécessite une expertise humaine en plus (M. Vladimir et al, 2000).

De nouvelles approches évolutionnaires sont actuellement utilisées pour la détection d'intrusion sur les réseaux et qui manipulent des structures beaucoup moins complexes et moins coûteuses que les autres approches de datamining qui traitent ce problème. La programmation génétique (Genetic Programming : GP) est une catégories des algorithmes évolutionnaires (AE) adopté par John Koza en 1992 (J.R.Koza, 1992). Les AE font évoluer une population d'individus, où seulement les mieux adaptés peuvent survivre et se reproduire. L'utilisation des GP s'est étendue pour la résolution de nombreux types de problèmes dont les solutions peuvent être

représentées par des structures arborescentes. Avec cette approche, on fera évoluer une population d'individus représentés sous formes d'arbres génétiques. Chaque individu est un ensemble de règles d'induction permettant de classer chaque ligne dans la base dans une des cinq catégories des intrusions Internet. Pour mettre au point un tel algorithme, il faut définir l'espace de recherche (les règles d'induction), la fonction objectif à optimiser ainsi que les opérateurs génétiques (croisement, mutation et sélection) pour l'implantation de l'évolution.

Ces algorithmes (D. Song et al, 2003) ont été utilisés pour résoudre beaucoup de problèmes de sécurité des systèmes informatiques. Dans cet article, nous présentons une nouvelle application de la programmation génétique pour résoudre les problèmes de détection des intrusions Internet. Cette approche se base sur une technique d'apprentissage à partir d'une base de donnée de connexions Internet. La base utilisée pour ce travail est la base KDD contenant un très grand nombre de connexions normales et des attaques de différents types.

3 La base d'apprentissage KDD

Vue l'importance des IDS dans l'ingénierie Informatique, et afin d'encourager les recherches dans ce domaine pour améliorer la performance des IDS, une équipe du laboratoire Lincon à l'institut MIT a mis au point en 1998/99 une base test KDD (K.Kendall, 1999) pour l'évaluation des IDS.
Cette base a permis de comparer la performance des IDS et les résultats de recherches dans le domaine en utilisant les mêmes données.

La base KDD est composée de 5 millions de connexions. Chaque connexion est décrite par 41 champs et un libellé indiquant si la connexion est normale ou une attaque tout en spécifiant le type de l'attaque. On compte 22 types d'attaques, regroupés en 4 catégories (D. Song et al, 2003) :

- Denial of Service (DoS)
- Remote to Local (R2L)
- User to Root (U2R)
- Probe

Normal	19.69%
Denial of Service (DoS)	79.24%
Remote to Local (R2L)	0.23%
User to Root (U2R)	0.01%
Probe	0.83%

Tableau1. *Distribution des attaques*

3.1 DOS (Denial of service)

Le déni de service est une classe d'attaque visant à rendre indisponible un service. Ceci peut s'effectuer de plusieurs manières : par le biais d'une surcharge réseau, rendant ainsi la machine totalement injoignable.

3.2 R2L (Remote to Local)

Cette classe d'attaque consiste à exploiter les vulnérabilités de la machine pour gagner illégalement l'accès local en tant qu'utilisateur.

3.3 U2R (User to Root)

Cette classe d'attaque consiste à accéder au compte normal utilisateur sur le système et exploiter la vulnérabilité pour gagner des privilèges de super utilisateur.

3.4 Probe

Cette classe d'attaque consiste à balayer un réseau pour recueillir des informations sur le système cible ou pour trouver des vulnérabilités connues.

Chaque connexion est décrite par 41 champs et un autre champ indiquant l'état de la connexion normal ou attaque. Les 9 premiers champs sont les champs de base de chaque connexion (D. Song et al, 2003). Les 32 champs restant sont classés en trois catégories :

- Champs contenu : La connaissance de domaine est employée pour évaluer la charge utile des paquets initiaux de TCP. Exemple de champs tel que le nombre de tentatives échouées de procédure de connexion.
- Champs trafic à base de temps : ces champs sont conçues pour capturer des propriétés qui mûrissent sur une 2 deuxième fenêtre temporelle. Un exemple de champ, le nombre de connexions au même centre serveur pendant l'intervalle de 2 secondes
- Champs trafic à base d'hôte : utilise une fenêtre historique évaluée sur le nombre de connexions - dans ce cas 100 - au lieu du temps. Ces champs sont donc conçus pour évaluer des attaques, qui recouvrent des intervalles plus longs que 2 secondes.

Champs	Description
Duration	Longueur (nombre de secondes) d'une connexion
protocol_type	Type de protocole (ex : TCP,UDP, etc)
Service	Service réseau en destination, ex: http, telnet, etc
Flag	Normal ou erreur de connexion
Src_bytes	Nombre de bit des données de la source à la destination
Dst_bytes	Nombre de bit des données de la destination à la source
Land	1- connexion de/à la même hôte ; 0- sinon
wrong_fragment	Nombre de fragment erroné
Urgent	Nombre de paquet urgent

Tableau 2. *Champs de base*

Champs	Description
Hot	Nombre d'indicateur « hot »
Num_failed_logins	Nombre d'échec essai de login
logged_in	1- connexion avec succès; 0-sinon
Num_compromised	Nombre de condition compromis
root_shell	1-root shell obtenu ; 0-sinon
su_attempted	1-« su root » commande obtenu ; 0- sinon
Num_root	Nombre d'accès à la racine
Num_file_creations	Nombre d'opération de création de fichier
Num_shells	Nombre de prompts shell
Num_access_files	Nombre d'opérations d'écriture, suppression et création dans l'accès au contrôle de fichiers
Num_outbound_cmds	Nombre de commande outbound dans la session ftp
is_host_login	1- login appartient à la liste « hot » (ex : root, adm, etc) ; 0- sinon
is_guest_login	1- le login est un login de visiteur (ex : guest, anonymous, etc) ; 0- sinon

Tableau 3. *Champs de contenu*

Vu la grande taille de la base en entrée, KDD'99 offre une base d'apprentissage constitué de 10% de la base d'origine (494021 enregistrements) en conservant les mêmes pourcentages de connections normal (19.69%) et attaques (80.31%). C'est sur cette base réduite que portent les travaux de cet article.

Cependant, la taille de la base reste encore énorme pour un programme génétique, d'où il est exclu de l'utiliser en entier pour l'étape de l'apprentissage.

Ainsi, il est nécessaire de passer d'abord par une étape de préparation permettant d'échantillonner les données. Les échantillons crées sont ensuite utilisés soit pour l'étape d'apprentissage soit pour l'étape de validation (test), ou pour les deux.

Le contenu et la taille des échantillons doivent être réglés afin d'optimiser l'étape d'apprentissage en terme temps de calcul et quantité d'information utile pour la procédure d'induction (classificateur). Pour notre travail, nous avons crée des échantillons de ~5000 enregistrements (1% de base) en conservant les mêmes pourcentages de connections normal et des quatre types attaques pour avoir des classes de données équilibrées.

4 Expérimentation

L'approche évolutionnaire qui a été adoptée dans notre travail est la programmation génétique. Nous avons utilisé les outils de développement des AE fournis par la bibliothèque C++ *EOlib* (M.Keijzer, 2002). EOlib a été créée par l'équipe Geneura à l'université de Grenada et a été renforcée par Marten Keijzer et Marc Schoenauer (M.Keijzer, 2002). EOlib implémente les bases des AE standard (algorithmes génétiques, stratéges d'évolution, programmation génétique).
Dans notre programme, nous avons implémenté en plus la représentation arborescente et le calcul de la fonction objective. Le nouveau code a été ensuite intégré dans le noyau de la plateforme EOlib.
Pour les expérimentations, nous avons préparé 11 échantillons pour l'apprentissage et un pour la partie test. Chaque échantillon est formé de ~5000 enregistrements et de 10 variables seulement pour réduire la taille des données. Nous avons réduit le nombre de variable pour réduire la taille des données.

Les paramètres du GP, choisi aléatoirement, implémenté sont les suivants:

- Nombre de génération=100,
- Taille population=50,
- Probabilité de mutation=0.1,
- Probabilité de croisement=0.6,
- Sélection par Tournoi.

Afin d'estimer l'efficacité de notre algorithme et d'évaluer l'impact des données d'apprentissage sur la qualité des résultats, nous avons utilisé deux techniques de sélection des données :

- Sélection des données d'apprentissage avec un seul échantillon : le principe de cette technique est d'utiliser un seul échantillon pour toutes les générations du programme génétique.
- Sélection des données d'apprentissage avec plusieurs échantillons : l'objectif de cette technique est le changement de l'échantillon d'apprentissage d'une façon aléatoire au cours de l'évolution. Elle consiste à tester si le nombre de génération avant de changer l'échantillon. Si oui, sélectionner d'une façon aléatoire un échantillon ensuite changer (parmi une liste) et changer l'échantillon de l'évolution. Dans ce type de

génération, nous avons utilisés deux nouvelles variables: nombre d'échantillon et le nombre de génération avant de changer l'échantillon.

- Nombre d'échantillon=11.
- Nombre de génération avant de changer l'échantillon=10.

Entre autres, nous avons choisi un seul échantillon test, l'échantillon 1, pour les deux techniques de sélection de données.

5 Résultat

Les résultats obtenus avec les deux techniques de sélection des données d'apprentissage sont décrits dans les deux sous sections suivantes.

5.1 Sélection des données d'apprentissage avec un seul échantillon

Le tableau ci-dessous présente les résultats obtenus pour 11 tests de notre algorithme. Chaque test a été effectué avec un des échantillons préparés comme décrit dans la section 3.

On remarque que l'algorithme arrive à bien classer toutes les attaques de types Dos (taux d'erreur égal à 0) et presque toutes les attaques de type Probe (seul l'erreur de l'échantillon 5 est différente de 0). Entre autres, les résultats obtenus pour les différents échantillons sont très proches. Le pourcentage de mauvaises classements est presque similaire pour les connections normales et les attaques de type L2R. Ceci peut être expliqué par le fait que les échantillons d'apprentissage ont été crées avec la même procédure en gardant les mêmes pourcentages des différents types de connections.

Test	**Catégorie**	**Erreur par catégorie**	**Erreur totale**	**Erreur Totale moyenne**
Test1	Normal	0.00465493	0.00485732	
	DOS	0		
	L2R	0.000202388		
	U2R	0		
	Probe	0		
Test2	Normal	0.00425015	0.00546448	
	DOS	0		
	L2R	0.00101194		
	U2R	0.000202388		
	Probe	0		
Test3	Normal	0.00445254	0.00586926	
	DOS	0		
	L2R	0.00121433		
	U2R	0.000202388		

	Probe	0		0.00563007
Test4	Normal	0.00445254	0.00607165	
	DOS	0		
	L2R	0.00141672		
	U2R	0.000202388		
	Probe	0		
Test5	Normal	0.00445254	0.00566687	
	DOS	0		
	L2R	0.00101194		
	U2R	0		
	Probe	0.000202388		
Test6	Normal	0.00182149	0.00445254	
	DOS	0		
	L2R	0.00242866		
	U2R	0.000202388		
	Probe	0		
Test7	Normal	0.00425015	0.00546448	
	DOS	0		
	L2R	0.00101194		
	U2R	0.000202388		
	Probe	0		
Test8	Normal	0.00425015	0.00607165	
	DOS	0		
	L2R	0.00161911		
	U2R	0.000202388		
	Probe	0		
Test9	Normal	0.0068812	0.00708359	
	DOS	0		
	L2R	0		
	U2R	0.000202388		
	Probe	0		
Test10	Normal	0.00425015	0.00566687	
	DOS	0		
	L2R	0.00121433		
	U2R	0.000202388		
	Probe	0		
Test11	Normal	0.00425015	0.00526209	
	Dos	0		
	L2R	0.00101194		
	U2R	0		
	Probe	0		

Tableau 6. *Résultats avec un seul échantillon*

5.2 Sélection des données d'apprentissage avec plusieurs échantillons

	Catégorie	Erreur par catégorie	Erreur totale
Test	Normal	0.00182149	0.00445254
	Dos	0	
	L2R	0.00242866	
	U2R	0.000202388	
	Probe	0	

Tableau 7. *Résultats avec plusieurs échantillons*

Comme pour les premières expérimentations, l'algorithme réussit à classer correctement toutes les attaques de type Dos et Probe.

5.3 Comparaison des résultats

Nous avons obtenus dans les deux types de sélections des erreurs très faibles. Entre autres, on remarques une légère amélioration de l'erreur moyenne qui passe de 0.56% avec en utilisant un seul échantillon de données d'apprentissage, à 0.44% avec plusieurs échantillons. Cette baisse est due essentiellement à l'augmentation du nombre des bons classements des connections normale qui passe de 99,56% dans le premier cas à 99,82% dans le deuxième cas. Cependant, cette amélioration reste faible par rapport au volume total des échantillons utilisés dans le deuxième cas.

6 Conclusion

Dans cet article, nous avons présenté une méthodologie de génération automatique d'un modèle décisionnel pour la classification des connections Internet utilisant un programme génétique de base.

Nous avons aussi testé deux types d'approches de sélection de données d'apprentissage de la base KDD. La première approche utilise un seul échantillon d'apprentissage aléatoire tout au long de l'évolution alors que la deuxième utilise un ensemble d'échantillons avec un choix aléatoire de l'échantillon d'apprentissage pendant un intervalle donné de l'évolution. Nous avons constaté dans l'étude faite que les résultats obtenus sont très proches en terme de temps de calcul et en terme des taux d'erreur obtenus, avec une légère amélioration de la qualité des résultats avec la deuxième technique.

Dans des travaux futurs, nous envisageons de développer d'autres approches de sélection des données comme des sélections adaptatives, dont l'objectif est d'enrichir l'échantillon d'apprentissage avec les types de connections les plus difficiles à classer tout au long de l'évolution. Nous allons étudier également l'impact des choix de variables de la base d'apprentissage dans la génération des IDS par la programmation génétique.

Références

D. Song, M. Heywood, and A.N. Zincir-Heywood « A Linear Genetic Programming Approach to Intrusion Detection », Genetic and Evolutionary Computation—GECCO, 2003, volume 2724 of LNCS, pages 2325- 2336, Chicago, 12-16 June, Springer.

I. Levin «KDD-99 Classifier Learning Contest LLSoft's Results Overview». SIGKDD Explorations. ACM SIGKDD. 1(2) -65-75, 2000.

J.R.Koza, « Genetic Programming: On the Programming of Computers by means of Natural Evolution », MIT Press, Massachussetts, 1992.

Kristopher Kandall, « A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems », Master's Thesis, Messachussetts Institue of Technology, 1999.

Kumar J. Das, « Attack Development for Intrusion Detection Evaluation », Messachussetts Institue of Technology, 2000.

M.Keijzer, J.Merelo, G.Romero, M. Schoenauer, « Evolving Objects: a general purpose evolutionary computation library»,URL: http://www.eolib.org, 2002.

M. Crosbie, P.G. Spafford, « Applying Genetic Programming to Intrusion Detection », *In Proceedings of 1995 AAAI Fall Symposium on Genetic Programming*, pp. 1-8. Cambridge, Massachusetts. http://citeseer.nj.nec.com/crosbie95applying.html (30 Oct. 2003), 1995.

M. Validimir , V Alexei, S. Ivan, «The MP13 Approach to the KDD-99 Classifier Learning Contest». SIGKDD Explorations. ACM SIGKDD. 1(2) -76-77, 2000.

P. Biondi, « Architecture expérimentale pour la détection d'intrusions dans in système d'informations ». Rapport technique http://www.secdev.org/ids.pdf, 2001.

S. Mukkamala, A. Sung, A. Abraham, « Modeling Intrusion Detection Systems Using Linear Genetic Programming Approach », *Innovations in Applied Artificial Intelligence*, 2004, Springer.

W.Lee,S.J.Stolfo «A Framework for Constructing Features and Models for Intrusion Detection Systems», *Proceedings of the IEEE Symposium on Security and Privac* , 1999 .

W.Lee, S.J.Stolfo, and K.W. Mok, «A Datamining Framework for Building Intrusion Detection Models», *Proceedings of the 1999 IEEE Symposium on Security and Privac*, May 1999.