Détection d'intrusions réseaux par l'utilisation de cartes de Kohonen et d'un réseau de Neurones

Florian BOITREL
Maxime CHAMBREUIL

Tuteur : M. Philippe Leray

25 juin 2003

Table des matières

1 Introduction, principe

1.1 L'origine du projet

Ce projet est issu de travaux réalisés par des scientifiques et informaticiens de l'Institut Polytechnique de Rensselaer à New York. L'idée de départ consiste à repérer les attaques réseaux (par exemple de type DOS : Denial Of Service) à partir des données du traffic réseau. Pour cela ils ont mis en place un système de détection d'anomalies réseaux qui permet d'alerter les administrateurs système en cas de traffic anormal.

1.2 Principe général

Les données sont issues de Tcpdump, un sniffeur réseau. Afin de traiter ces données, une carte de Kohonen (SOM : Self-Organizing Map) et un réseau de neurones de type MLP (Multi-Layered Perceptron) sont utilisés. Un pré-traitrement est nécessaire sur les données de Tcpdump, comme nous le verrons dans la section Données.

1.3 La carte de Kohonen

La carte de Kohonen permet de créer des groupes de sources. Plus clairement, en sortie de la carte de Kohonen nous avons des classes avec des comportements différents selon le nombre de connexions par port. Ce type de regroupement est nécessaire car dans un certain intervalle de temps une machine peut être source d'attaque et source normale dans un autre intervalle de temps. En outre il peut y avoir plusieurs sources en communication avec la victime, donc un simple classement par machine cible n'est pas satisfaisant. Les traitements sont effectués tous les Δt intervalles de temps.

1.4 Le perceptron multi-couches

Le réseau de neurone prend en entrée les données issues de la carte de Kohonen. Suite à un apprentissage, il permet de détecter si une attaque a eu lieu ou pas. Nous verrons qu'il est plus aisé de détecter un type d'attaque bien précis (en connaissant son principe d'attaque, i.e. par exemple le nombre de connections sur tel port en tant de temps) que l'ensemble de toutes les attaques possibles.

2 Données

2.1 Tcpdump

Dans notre cas ces données sont issues de tcpdump ; il s'agit d'un sniffeur réseau qui donne entre autres :

1. les adresses IP des machines sources

2. les adresses IP des machines cibles
3. le moment (temps) de réception du paquet
4. le port de destination
5. le port source
6. le protocole utilisé

Dans notre cas nous avons besoin du temps, des adresses sources et de destination et des ports de destination.

2.2 Exemple de données issues de tcpdump

```
15:00:22.535006 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
15:00:23.534934 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
15:00:24.534133 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
15:00:25.534535 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
15:00:26.484704 192.168.1.1.router > RIP2-ROUTERS.MCAST.NET.
router:  RIPv2-resp [items 1]: {172.16.0.0/255.255.0.0}(1) [tos 0xc0]
15:00:27.035525 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
15:00:28.034163 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
15:00:29.034660 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
15:00:30.035005 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
15:00:31.034976 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
15:00:31.487154 0:10:7b:38:46:32 0:10:7b:38:46:32 loopback 60:
                        0000 0100 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000
15:00:31.906412 arp who-has 192.168.1.90 tell 192.168.1.10
15:00:31.906654 arp reply 192.168.1.90 is-at 8:0:20:11:34:bf
15:00:33.034354 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
15:00:34.036749 8:0:9:61:aa:c9 netbeui > 8:0:9:61:aa:c9 null
test/P len=37
```

Au vue de cet échantillon, on se rend bien compte de la difficulté d'exploitation que présente ce format de données.

3 Carte de Kohonen

3.1 Sélection de ports

La durée d'observation du traffic est de $F \times \Delta t$, F étant un simple coefficient multiplicateur. Au départ, quelques ports de base (ici par exemple trois) sont définis par l'administrateur système. Ces ports sont en général des ports "classiques", par exemple les ports 21,22 et 23. Deux ports supplémentaires sont générés, ce sont ceux qui ont le nombre de connexions maximales.

Un exemple de sélection de ports (extra ports : 25 et 80) :

Ports Given

21
22
23

Extra Ports

ep_1
ep_2

+

Activity during F * Δt

Ports	Hits
21	40
20	17
22	17
23	15
25	24
80	49

=

FINALSET

21
22
23
25
80

FIG. 1 – Comment trouver les ports finaux d'étude

3.2 Principe de la carte de Kohonen

	Time	Source	Dest. Port
first Δt	92635327	10.1.9.2	80
	92635328	10.1.9.2	20
	92635328	10.1.9.5	23
	92635331	10.1.6.2	21
	92635334	10.1.9.5	161
	92635338	10.1.6.2	21
	92635342	10.1.6.2	80
	92635344	10.1.9.2	21
	92635349	10.1.9.5	23
	92635354	10.1.9.5	21
Δt *Separator*			
second Δt	92635360	10.1.9.2	80
	92635360	10.1.9.5	20
	92635361	10.1.6.2	80
	92635362	10.1.6.2	21
	92635363	10.1.9.2	161
	92635365	10.1.9.5	21
	92635371	10.1.9.5	23
	92635373	10.1.6.2	80
	92635373	10.1.9.2	21
	92635375	10.1.6.2	25
	92635379	10.1.9.2	21
Δt *Separator*			

FIG. 2 – Données issues de tcpdump prétraitées

Le principe de la carte de Kohonen, au niveau de l'implémentation est de récupérer les quatre meilleurs BMU (ou Best Matching Unit).

	Source	Port 21	Port 22	Port 23	Port 25	Port 80
first Δt	10.1.9.2	1	0	0	0	1
	10.1.9.5	1	0	2	0	0
	10.1.6.2	2	0	0	0	1
Δt *Separator*						
second Δt	10.1.9.2	2	0	0	0	1
	10.1.9.5	1	0	1	0	0
	10.1.6.2	1	0	0	1	2
Δt *Separator*						

FIG. 3 – Les données précédentes groupées en fonction des 5 ports finaux choisis

	Source	Port 21	Port 22	Port 23	Port 25	Port 80
first Δt	10.1.9.2	8	0	50	11	220
	10.1.9.5	54	3	63	132	687
	10.1.6.2	0	37	0	0	0
	10.1.6.5	0	0	13	0	54
Δt Separator						
second Δt	10.1.9.2	0	0	120	0	0
	10.1.5.3	154	23	0	0	0
	10.3.6.5	0	0	60	0	286
	10.3.9.8	76	0	0	187	0
	10.3.9.17	0	0	73	0	321
	10.4.16.64	0	112	0	0	0
Δt Separator						

FIG. 4 – Totaux par port avant entrée dans la carte de Kohonen

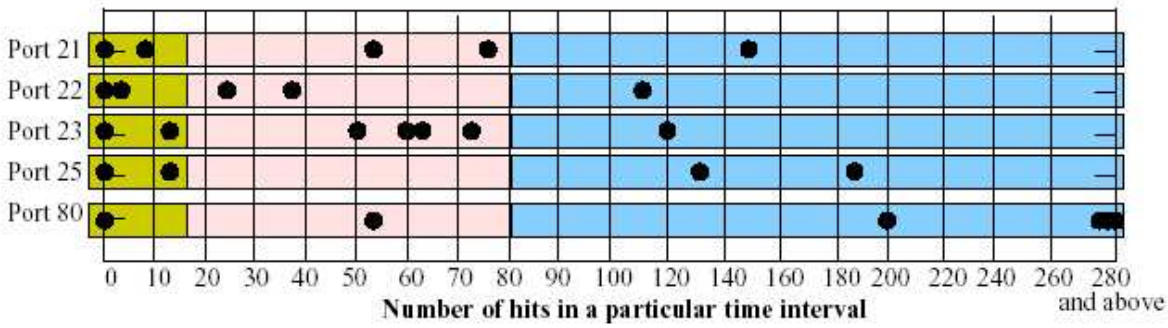

FIG. 5 – Exemple du nombre de connexion par ports selon le temps

4 Prise de décision avec le réseau de neurones

4.1 Données en entrée du réseau de neurones

	Cluster #	Port 21	Port 22	Port 23	Port 25	Port 80
first Δt	Cluster 1	8	0	50	11	220
	Cluster 2	54	3	63	132	687
	Cluster 3	0	0	0	0	0
	Cluster 4	0	37	13	0	54
Δt Separator						
second Δt	Cluster 1	154	23	120	0	0
	Cluster 2	76	0	60	187	286
	Cluster 3	0	0	73	0	321
	Cluster 4	0	112	0	0	0
Δt Separator						

FIG. 6 – Exemple de classificatione en sortie de la carte de Kohonen

4.2 Configuration optimale

Après de nombreux tests effectués par l'équipe scientifique, il s'avère que le réseau de neurone optimal est celui décrit ci-dessous. Nous avons utilisé les mêmes configurations pour nos tests. Réseau de neurones avec le plus petits nombre de noeuds et le meilleur taux d'erreur :

- 4 ports de base ont été spécifiés et 1 port supplémentaire
- Le MLP possède en entrée 4 sources de comportement avec chacune 5 ports, donc il y a un total de 20 neurones en entrée
- Le MLP a une couche cachée de 25 neurones.
- Le MLP a une couche de sortie linéaire avec 1 seul neurone.

L'algorithme de rétro-propagation avec early-stopping utilisé comporte un maximum de 10000 itérations. L'apprentissage s'effectue avec 100 échantillons et est testé sur 50 échantillons. L'apprentissage comporte autant de cas d'attaques que de cas normaux. Chaque neurone du MLP comporte une seule valeur et le neurone de sortie peut prendre deux valeurs, 50 pour une attaque, 0 pour un comportement normal.

4.3 Structure du MLP

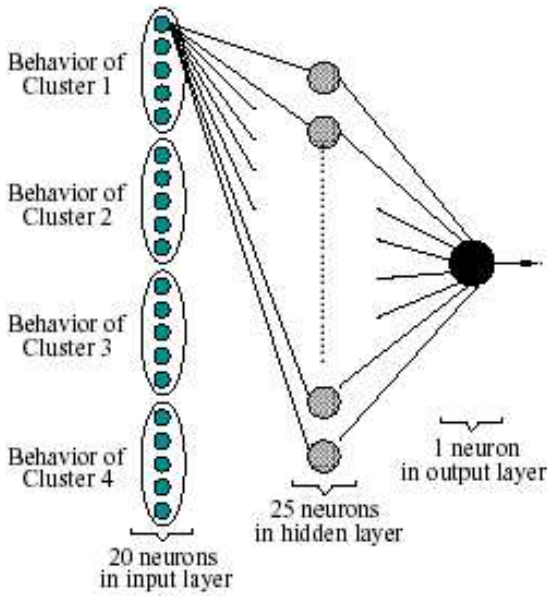

FIG. 7 – Structure du réseau de neurones MLP

5 Travail réalisé

5.1 Déroulement du projet

Dans un premier temps, nous avons lu l'article fourni par M. Leray et nous avons récupéré des jeux de données. Lorsque nous avons découvert les formats des données, ils n'étaient pas exploitables et la programmation d''un prétraitement aurait pris trop de temps. Nous avons donc contacté les auteurs, qui avaient déjà des résultats, pour récupérer leur programme de prétraitement et demander de plus amples explications. Ils nous ont alors renvoyé sur une thèse qui a été faite sur le même sujet.

En attendant leur programme de prétraitement, nous avons cherché des logiciels qui pouvait faire cette tâche (Tcptrace entres autres). Nous en avons trouvé et testé quelques-uns mais aucun ne nous a satisfait.

Parallèlement à cela, nous avons commencé à programmer (en Matlab) la carte de Kohonen et le perceptron multi-couche comme décrit dans la thèse. Nous avons rencontré quelques incohérences dans le fonctionnement global de l'application, c'est pourquoi nous avons modifié l'architecture proposée pour obtenir de meilleurs résultats.

En générant des données manuellement, nous sommes capable de détecter une attaque sur un port spécifique, comme cela a déjà été décrit.

5.2 Difficultés rencontrées

5.2.1 La préparation des données

Durant ce projet, la charge de travail générée par la préparation des données nous a posé un réel problème. C'est pourquoi nous avons demandé très tôt le programme de prétraitement des auteurs. Malheureusement, l'auteur du programme n'est plus dans l'université et il a essayer en vain de recontacter ses professeurs pour récupérer son programme.

Durant ce semestre, l'un de nous a suivi l'UV livre "Diagnostique du réseau INSA", qui avait pour objectif de faire de la métrologie réseau. Nous avons tenté de profiter de leurs expériences, mais ils ont eu les mêmes problèmes que nous et en sont arrivés à la conclusion qu'il valait mieux utiliser les facultés statistiques du logiciel de capture ou faire nos propres captures, afin de les stocker sous un format exploitable. Par ailleurs, ils ont rencontré les mêmes difficultés que nous à utiliser des logiciels d'exploitation de données tcpdump.

5.2.2 Enchainement des phases

Nous n'avions pas très bien compris, dans l'article et dans la thèse, l'enchainement des actions entre l'apprentissage de la carte de Kohonen, celui du MLP, la phase de test, etc...
Nous avons donc testé plusieurs solutions :

Dans un premier temps, nous sommes partis sur une base temporelle. Nous générions un échantillon tous les Δt. Les 100 premiers échantillons servaient à l'apprentissage et les 50 suivants aux tests et à la détection. Puis on recommencait le cycle "apprentissage + détection".

Par la suite, M. Leray nous a proposé différentes façons d'arrêter la phase de détection et de reprendre l'apprentissage (au lieu de faire ça par rapport au temps). Mais les solutions proposés concernaient la carte de Kohonen. Or nous avons compris par la suite que la carte est toujours utilisée pour de l'apprentissage : on projette sur la carte pour récupérer les BMUs, pas pour savoir sur quel neurone est projeté une donnée.

Enfin, après avoir cherché différentes méthodes d'arrêt de la détection, nous en avons déduit que la base temporelle n''était pas une une si mauvaise solution car l'utilisation d'un réseau est périodique dans le temps (différence de traffic jour/nuit par exemple). Par ailleurs, il est difficile de détecter et relance un réapprentissage si on obtient consécutivement N fois la même sortie.

6 Résultats

Avec la solution de l'article, on détecte bien une attaque sur un seul port avec un nombre de connexions élevé :

```
ID: 44.164944
Date: 04/01/1999
Name: New_Attack_1999 (sshprocesstable)
Category: dos
Start_Time: 16:49:15
Duration: 00:08:21
Attacker: 172.016.118.020
Victim: 172.016.112.050
Username: n/a
Ports:
        At_Attacker:
        At_Victim: 22{490}
```

En revanche, il existe tellement d'attaques qu'il nous est impossible de détecter toutes les attaques simultanément. En outre, il est difficile de détecter une attaque qui comporte peu de connexions sur 2 ports dans un laps de temsp très court (cf caractéristique d'attaque ci-dessous).

```
ID: 41.084818
Date: 03/29/1999
Name: sendmail
Category: r21
Start_Time: 08:48:12
Duration: 00:00:02
Attacker: 202.049.244.010
Victim: 172.016.114.050
Username: n/a
Ports:
        At_Attacker: 113{1}
        At_Victim: 25{1}
```

Concernant l'enchaînement des différentes phases, les 2 solutions donnent des résultats équivalents mais la solution temporelle nous parait être plus logique, car on s'adapte à la période de la journée. Par ailleurs, nous n'avons pas trouvé de critères de réapprentissage très satisfaisants.

7 Conclusion

Notre programme nous permet de détecter une attaque spécifique du type "sshprocesstable" à 99% avec 100 échantillons d'apprentissage et 300 de test.

Ce projet était intéressant de par l'utilisation d'une carte de Kohonen pour distinguer des sources aux comportements identiques. En phase de test du MLP, la détection d'attaque ne nécessite pas un temps de calcul très important et peut s'effectuer en temps réel. Ceci répond bien aux besoins d'un administrateur système.

Ce projet nous a permis de découvrir tous les problèmes liés à la préparation des données pour effectuer du traitement. Malheureusement, nous n'avons toujours pas reçu à l'heure actuelle les programmes des auteurs pour le faire

Une suite possible à ce projet pourrait être de préparer les données pour une mise en production et de tester différents ports pour ne surveiller que les plus probables pour une attaque. Il existe actuelment un besoin énorme de la part des administraterus et l'offre n'est pas très satisfaisante. C'est pourquoi beaucoup de scientifiques se penchent sur ce problème, ce qui fait l'obje de nombreuses thèses ou projet de recherche. Par ailleurs, de nouveaux types d'attaques apparaissent quotidiennement...

8 Sources

8.1 Article du projet

Network-based intrusion detection using neural networks
http://www.cs.rpi.edu/~szymansk/papers/annie02.pdf

8.2 Sites web

DARPA intrusion Detection Evaluation (Lincoln Laboratory, MIT)
http://www.ll.mit.edu/IST/ideval/data/data_index.html
Page contenant des données où des attaques ont été détectées :
http://www.ll.mit.edu/IST/ideval/data/1999/training/week2/index.html
Une base de données décrivant diverses attaques :
http://www.ll.mit.edu/IST/ideval/docs/1999/attackDB.html
Liste d'attaques décrites de façon plus détaillée, qui donne le nombre de connexions sur tel port en un certain laps de temps.
http://www.ll.mit.edu/IST/ideval/docs/1999/master_identifications.list
Le site de Tcpdump :
http://www.tcpdump.org
Le site de Tcptrace :
http://www.tcptrace.org

Thèse de John Alan Bivens (bivenj@cs.rpi.edu), Distributed framework for deploying machine learning in network management and security.