Adaptive Distributed Intrusion Detection Using Parametric Model

Jun Gao，Weiming Hu，Xiaoqin Zhang，and Xi Li
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
E-mail :{jgao, wmhu, xqzhang, lixi}@nlpr.ia.ac.cn

Abstract

Due to the increasing demands for network security, distributed intrusion detection has become a hot research topic in computer science. However, the design and maintenance of the intrusion detection system (IDS) is still a challenging task due to its dynamic, scalability, and privacy properties. In this paper, we propose a distributed IDS framework which consists of the individual and global models. Specifically, the individual model for the local unit derives from Gaussian Mixture Model based on online Adaboost algorithm, while the global model is constructed through the PSO-SVM fusion algorithm. Experimental results demonstrate that our approach can achieve a good detection performance while being trained online and consuming little traffic to communicate between local units.

1. Introduction

Over the past two decades, most algorithms for intrusion detection (ID) derive from the field of artificial intelligence, such as the statistics-based approaches, the data mining related approach, the neural networks and clustering methods. However, the aforementioned approaches have the following limitations. First, they are not directly applicable to data stream processing since these methods are trained offline. Second, they do not have the abilities of distributed processing. Compared with the centralized-based IDS, the distributed-based IDS is more popular and efficient in real world.

In general, there are three main challenges that must be addressed in distributed IDS. First, as a result of privacy protection, the communication between local units should shield the original data containing the privacy information. Second, in order to reduce the occupancy of network bandwidth, the communication traffic between local units should be as little as possible under the premise of keeping enough information to gain a global model. Third, how to effectively and efficiently combine all local models for improving the detection performance.

There are several meaningful work about online and distributed ID so far. Lee et al [1] combine the online clustering algorithm ART with Concept Vector and Mercer-Kernel. Thus this algorithm is unsuitable for the high-capacity data streams because the model parameters are lack of stability. Otey et al [2] implement the online algorithm based on frequent itemset mining and further propose a general-purpose distributed outlier detection algorithm, but their algorithm consumes the huge memory to store the features of training data.

In this paper, we present a distributed IDS framework based on the MAdaboost and PSO-SVM algorithms to address the above challenges. We actualize the online training and high efficiency fusion of the local models, while reducing the communication traffic between local units as much as possible by a parametric model based on Gaussian Mixture Model (GMM).

The remainder of this paper is organized as follows. Section 2 introduces the framework for distributed intrusion detection. Section 3 reports the experimental results, and Section 4 concludes the paper.

2. Framework of our distributed IDS

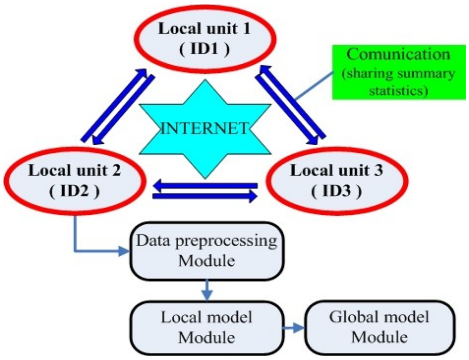

Figure 1 Framework of our distributed IDS

As shown in Figure 1, the framework of our distributed IDS consists of the following three modules:

- **Data preprocessing:** A set of data has to be labeled for training. They should contain both normal samples labeled as '+1' and attack samples labeled as '-1,-2…'. Then the following three groups of features are extracted: basic features, content features, and traffic features, as introduced in [3].

- **Local model:** The individual model for each local unit is constructed based on the online Adaboost and Expectation-maximization (EM) algorithms. The weak classifiers required by the Adaboost algorithm are constructed based on GMM.
- **Global model:** After the global broadcast, all local models are combined together to form the global model based on PSO and SVM algorithms. Then, local units can construct a cascade classifier based on the global and local models to detect the intrusion or just use the fusion classifier based on the global model.

2.1. Local model module

A. Multiple Adaboost algorithm

Our multiple Adaboost algorithm (MAdaboost) is the multiple-updating algorithm, which can update several ensemble members simultaneously by using a training sample.

The training sample is (x, y), where $y \in \{+1,-1,\ldots\}$. $H=\{h_t \mid t=1,\ldots T\}$ is a set of weak classifiers. $\{S^+, S^-, SC, C_t, \lambda_t^{SC}, \lambda_t^{SW}\}$ are the parameters in the training processes, where $\{S^+, S^-\}$ are respectively the numbers of the trained positive samples and negative samples; SC is the number of samples correctly classified by the strong classifier, compared with C_t for weak classifier h_t. There are six steps in the MAdboost algorithm:

1. Initialization.
$$\begin{cases} \lambda = (S^+ + S^-)/S^+ & if\ y=1 \\ \lambda = (S^+ + S^-)/S^- & else \end{cases} \quad \lambda^* = \lambda \tag{1}$$
2. Sort $H=\{h_{r1}, h_{r2},\ldots, h_{rT} \mid ri \in \{1,2,\ldots,T\}\}$ by $\{fusion\_\varepsilon_t\}$ in ascending order for the next process.
$$\varepsilon_t = \lambda_t^{sw}/(\lambda_t^{sc} + \lambda_t^{sw}) \tag{2}$$
$$fusion\_\varepsilon_t = (1-\alpha)*\varepsilon_t - \alpha*sign(y)*h_t(x) \tag{3}$$
3. Update $\{h_{ri}\}$ with $fusion\_\varepsilon_{ri} \le 0.5$, according to the order in $H=\{h_{r1}, h_{r2}, \ldots h_{rT}\}$.
Compute the iterations P_{ri} for h_{ri}
$$P_{ri} = P*\exp[-\gamma*(fusion\_\varepsilon_{ri} - \min\_\varepsilon)] \tag{4}$$
where $min\_\varepsilon = min\{fusion\_\varepsilon_{ri}\}$.
Start the loop for h_{ri}:
   i. Set τ according to $Poisson(\lambda)$, and update h_{ri} based on the *Learn* algorithm (see *B. Weak classifiers*) using the training sample τ times.
   $$h_{ri} = Learn(\tau,(x,y))$$
   ii. Compute λ, λ_{ri}^{SC}, λ_{ri}^{SW}
   $$\begin{cases} \lambda_{ri}^{sc} = \lambda_{ri}^{sc} + \lambda \\ \lambda = \lambda*\left(\dfrac{1-2\alpha}{2*(1-fusion\_\varepsilon_{ri})}\right), & if\ sign(y) = h_{ri}(x) \end{cases} \tag{5}$$
   $$\begin{cases} \lambda_{ri}^{sw} = \lambda_{ri}^{sw} + \lambda \\ \lambda = \lambda*\left(\dfrac{1+2\alpha}{2*fusion\_\varepsilon_{ri}}\right), & if\ sign(y) \ne h_{ri}(x) \end{cases} \tag{6}$$
4. Update the $\{\lambda_t^{SC}, \lambda_t^{SW}\}$ of $\{h_t\}$ with $fusion\_\varepsilon_t > 0.5$.
$$\begin{cases} \lambda_t^{sc} = \lambda_t^{sc} + \lambda^* & if\ sign(y) = h_t(x) \\ \lambda_t^{sw} = \lambda_t^{sw} + \lambda^* & else \end{cases} \tag{7}$$
5. Compute SC using the past ensemble weights $\{\rho_t\}$ before updating H with the current sample, and compute the parameters $\{C_t\}$.
6. The strong ensemble classifier :
$$H(x) = sign\left(\sum_{t=1}^{T} \rho_t * h_t(x) - \varsigma\right) \tag{8}$$
where
$$\rho_t = \rho_t^* \Big/ \sum_{t=1}^{T} \rho_t^* \quad , \quad \varepsilon_t = \lambda_t^{sw}/(\lambda_t^{sc} + \lambda_t^{sw})$$
$$\rho_t^* = \beta * \log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) + (1-\beta) * \log\left(\frac{C_t}{SC}\right) \tag{9}$$

The ensemble weights $\{\rho_t\}$ are composed of ε_t and $\log(C_t/SC)$, which are different from the ones in the traditional Adaboost algorithm. $\log(C_t/SC)$ called "Contributory Factors" means the contribution rate of h_t to the strong classifier, and can tune the ensemble weights to attain the better detection performance. Attention is required that ρ_t^* equal to zero if $\rho_t^* < 0$.

ς is the threshold for the strong classifier, which is determined by the average of the output values of a fixed window or empirically.

B. Weak classifiers

The weak classifiers, which are inputs of Adaboost algorithm, can be linear classifiers, Artificial Neural Networks (ANNs) or other common classifiers. In order to decrease the communication traffic between local units as much as possible, we construct the weak classifiers according to the GMM. The GMM can be described by several parameters, which means that the global broadcast only needs to include a set of summary parameters rather than all training samples.

Suppose the number of features from training data is D, then there are D weak classifiers for the MAdaboost algorithm. For the behaviors labeling c, the GMM on the *jth* feature is:
$$\theta_j^c = \left\{\omega_j^c(i), \mu_j^c(i), \sigma_j^c(i)\right\}_{i=1}^{K} \tag{10}$$
where $j \in [1,D]$ and $c \in \{+1,-1,-2,\ldots\}$ is the labels of the behaviors, in which the normal is labeled as +1.

Then, the weak classifier on the *jth* feature is:
$$h_j(x) = sign(\arg\max_c \{\varphi_c(x)\}) \tag{11}$$
$$\varphi_c(x) = \begin{cases} p(x \mid \theta_j^c) & c = 1 \\ p(x \mid \theta_j^c)/W & c \ne 1 \end{cases} \tag{12}$$
where W is the total class number of intrusions and can balance the importance of positive and negative samples as the weight of the conditional probabilities.

The parameters of the GMM can be obtained

through the EM or K-means algorithm. We use the *Learn* algorithm based on the sequential EM algorithm to solve this problem, which updates θ_j^y for τ times using a training sample (x, y). Limited to the length of the article, the details about this algorithm is showed in [4]. Attention is required that the *Learn* algorithm needs not call the weighted incremental PCA algorithm.

2.2. Global model module

Though the Local model module, we can gain the local model for every local unit:

$$\psi = \{\psi_\rho, \psi_\theta, \varsigma\} \tag{13}$$

where $\psi_\rho=\{\rho_i \mid i\in[1,D]\}$, ρ_i is the ensemble weight for the *ith* weak classifier; $\psi_\theta=\{\theta_j^c \mid c\in\{+1,-1,-2, \ldots\}, j\in[1,D]\}$, where θ_j^c is the parameters of the GMM; ς is the threshold for the strong classifier.

We combine the Particle Swarm Optimization (PSO) and SVM algorithms to fuse the local models. By combining the strong searching ability of PSO and the small sample learning ability of SVM, the local units can construct the global model just based on the small sample as fast as possible. The PSO-SVM pseudo-code is shown in Table 1.

Table 1 PSO-SVM fusion algorithm

Initialize:

$\{X_{i,0}\}_{i=1}^M$ (Randomly be chosen in particle space)

$\{P_{li} = X_{i,0}\}_{i=1}^M \quad P_g = \arg\max_{P_{li}} f(P_{li})$

Loop:

1. If $f(P_g)>max\_fintness$ or iterations reaches the threshold value, exit.
2. Construct the SVM classifier respectively for each particle $X_{i,n}$, and calculate the detection rate $\gamma(X_{i,n})$.
3. Update $\{f(X_{i,n})\}_{i=1}^M$.
4. Update $\{P_{li}\}_{i=1}^M$ and P_g .
5. Update $\{V_{i,n+1}, X_{i,n+1}\}_{i=1}^M$.

End

Construct the ultimate SVM classifier for P_g.

Suppose the number of local units is N. Construct a vector as $(r_1, r_2,\ldots r_N)$, where r_i is the result of the *ith* local unit for the current data. Attention is required that these results are in the range [-1, 1]:

$$r_i = \sum_{t=1}^{T} \rho_t * h_t(x) \quad (\varsigma = 0) \tag{14}$$

Limited to the length of the article, the details about PSO is showed in [5]. Compared with traditional PSO, there are two differences needed to attend.

- The calculation of velocity :

$$\begin{aligned} V_{i,n+1} &= F\left(wV_{i,n} + c_1 rand()(P_{li} - X_{i,n}) + c_2 Rand()(P_g - X_{i,n})\right) \\ X_{i,n+1} &= X_{i,n} + V_{i,n+1} \end{aligned} \tag{15}$$

where $F()$ is a function to confine the velocity within a reasonable range: $\|V_{i,n}\| \le V_{max}$.

$$w = (w - 0.4) * (Titer - Iter) / Titer + 0.4 \tag{16}$$

where *Titer* is the maximum iteration number and *Iter* is the current iteration.

- The fitness value is evaluated as below:

$$f(X_{i,n}) = \nu * \gamma(X_{i,n}) + (1-\nu) * (|A| - |L|)/|A| \tag{17}$$

where $\gamma(X_{i,n})$ is the detection rate of the classifier based on the SVM algorithm for the particle $X_{i,n}$; A is the number of all local units, and L is the number of local units chosen by the particle $X_{i,n}$.

When the certain conditions are met, local models would globally broadcast their own local models. Then, each unit can construct the global model according to its own needs. If local units need the uniform global model, in the communication between all units, the shared information should include a small data sample besides the summary parameters. This sample can be constructed by randomly sampling from the local training data according to the proportion of various kinds of the network behaviors. If local units need the customized global model, the training data set would be obtained just by sampling from its own training data.

Once local units gain their own global model, the intrusions can be detected as follows:

1. Use the local models included in the global model to detect the current data, and obtain the result vector $[result_1, result_2, \ldots result_L]$, L is the length of the global best particle P_g.
2. Use the ultimate classifier (cascade or global classifier) to detect the current data.

3. Experiments

We utilize the KDD CUP 1999 data set which is condensed for IDS researches from DARPA. Four general types of attacks are defined in this data set: DOS (denial of service), U2R (user to root), R2L (remote to local) and PROBE (surveillance).

In our experiments, the parameters are set as follows: α=0.1, β=0.8, P=20, ς=0. In the following, we first show the results with different γ, and then compare the performance of our MAdaboost algorithm with those of the existing algorithms, and finally compare the performance of our PSO-SVM algorithm with that of fusion sum rule and SVM algorithm.

3.1. MAdaboost algorithm

As shown in Table 2, when γ ranges from 10 to 50, we can find that the moderate attenuation coefficient is important to the performance of the MAdaboost algorithm. If γ is too small, the training data are equivalent to being used to train all weak classifiers equally; if γ is too large, the training data are equivalent to only being used to update the weak classifier with the minimal $fusion\_\varepsilon_t$. When $\gamma\in[20,30]$, we construct the better grade for the updating times of all weak classifiers.

Table 3 shows the performances of some existing algorithms. Compared with the offline algorithms, our algorithm not only gains the satisfactory detection rate while keeping the lower false positive rate, but also can adaptively modify the local model in a real time manner. Compared with the online algorithm, our algorithm gains the preferable performance, especially on the lower false positive rate.

Table 2 Results of different γ

γ	FPR(%)	DR(%)
10	12.87	92.50
20	1.17	90.61
25	1.69	91.15
30	1.26	90.55
40	0.37	88.28
50	0.34	24.33

Table 3 Results comparison for local units

Methods		FPR(%)	DR(%)
Offline	Hierarchical SOM [6]	2.19-3.99	90.94-93.46
	Bagged C5 [7]	0.55	91.81
	Improved Adaboost [8]	0.31-1.79	90.04-90.88
Online	Mercer kernel ART[1]	2.9-3.4	92-95
	Our Method MAdaboost	**1.17-1.69**	**90.61-91.15**

3.2. PSO-SVM algorithm

In these experiments, we simulate the distributed IDS with 6 local units. For the PSO-SVM algorithm, we used the following training sets for local models, which only contain four low level kinds of attacks: neptune, smurf, portsweep, and satan. The number of these four kinds takes up 98.46% of the number of all kinds of attacks from 10% training set of KDD CUP 1999. The training set used for the fusion algorithms only contains 4000 randomly chosen records, and the testing sets for local and global models are the same, which contain 284672 samples of above four kinds of attacks and the normal kind of the network.

Table 4 shows that our combining algorithm greatly improves the performance of the classifiers, and is superior to the sum rule and SVM algorithm. Obviously, the performance disparities between different local models indicate that the sum rule isn't suitable for the distributed IDS. When the number of local units increases, the SVM algorithm used to combine all local models would not only consume huge time and resources, but also couldn't choose the best local model combination to improve the performance. Through dynamically combining a small portion of all local models to obtain the global model, our PSO-SVM algorithm effectively solves these problems, achieves the better performance, and simultaneously reduces the time consumption for detecting the intrusions.

Table 4 Results for distributed IDS of 6 units

Local models		FPR(%)	DR(%)
No.	Kinds of attacks		
1	neptune	0.0825	26.48
2	smurf	0.0017	70.16
3	portsweep	0.1782	7.92
4	satan	0.0083	0.81
5	neptune, smurf	0.1997	99.54
6	portsweep, satan	1.8154	26.77
Global model (PSO-SVM)		**0.3713**	**99.99**
Sum Rule		0.0066	26.37
SVM		0.3944	99.98

4. Conclusion

In this paper, we have introduced a adaptive distributed IDS framework based on the MAdaboost and PSO-SVM algorithms, which can achieve the preferable performance compared with other offline and online algorithms. In future, we will conduct some research on the parameters combining for distributed IDS framework to gain the better combining performance.

Acknowledgment

This work is partly supported by NSFC (Grant No. 60825204, 60672040) and the National 863 High-Tech R&D Program of China (Grant No.2006AA01Z453).

References

[1] H.Lee, Y.Chung, and D.Park, "An adaptive intrusion detection algorithm based on clustering and kernel-method", *Int. Conf. Adv. Inf. Netw. Appl.*, 2004, pp. 603-610.

[2] M.E.Otey, A.Ghoting, and S. Parthasarathy, "Fast distributed outlier detection in mixed-attribute data sets", *IEEE Trans. on Knowledge and Data Engineering*, May 2006, v12: 203-228.

[3] W.Lee, S.J.Stolfo, and K.Mok. "A framework for constructing features and models for intrusion detection systems", *ACM Trans. on Information an System Security*, November, 2000, 3(4):227-261.

[4] Lei.Y, Ding X Q, Wang S J. "Visual Tracker Using Sequential Bayesian Learning: Discriminative, Generative and Hybrid", *IEEE Trans. on Systems, Man and Cybernetics*, Part B, Dec. 2008, 38(6):1578-1591.

[5] J.Kennedy and R. Eberhart. "Particle swarm optimization", In *Proceedings of IEEE International Conference on Neural Networks*, 1995, volume 4:1942–1948.

[6] S.T.Sarasamma, Q.A.Zhu, and J.Huff. "Hierarchical kohonenen net for anomaly detection in network security", *IEEE Trans. on Systems, Man and Cybernetics*, Part B, April 2005, 35(2): 302-312.

[7] B.Pfahringer, "Winning the kdd99 classification cup: Bagged boosting", *SIGKDD Explorations*, 2000, 1(2): 65-66.

[8] W. M. Hu and W. Hu, "Adaboost-based algorithm for network intrusion detection," *IEEE Trans. on Systems, Man and Cybernetics*, Part B, April 2008, 38(2):577-583.

[9] Y.G. Wang, Xi Li, and W.M. Hu, "Distributed detection of network intrusions based on a parametric model", *IEEE Int. Conf. Syst., Man, and Cyber.*, Oct. 2008.