Network-based Intrusion Detection Using Adaboost Algorithm

Wei Hu and Weiming Hu
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
Email:{whu,wmhu}@nlpr.ia.ac.cn

Abstract

Intrusion detection on the internet is a heated research field in computer science, where much work has been done during the past two decades. In this paper, we build a network-based intrusion detection system using Adaboost, a prevailing machine learning algorithm. The experiments demonstrate that our system can achieve an especially low false positive rate while keeping a preferable detection rate, and its computational complexity is extremely low, which is a very attractive property in practice.

KEY WORDS
Intrusion detection, Network-based IDS, AdaBoost, Computational complexity

1 Introduction

With the development of the internet, the information security threat is becoming one of the most crucial problems. Reliable connections, information integrity and privacy are demanded more intensively nowadays than ever before.

Intrusion detection on the internet is all the while a heated field in computer science since its initiation by Denning [2] in 1987, in which a grand amount of research has been done. Generally speaking, intrusion detection systems (IDS) can be divided into two categories: host-based IDS and network-based IDS [1]. Host-based IDS utilizes various audit data of the target host machine. It has an advantage that the information provided by the audit data can be extremely comprehensive and elaborate. Network-based IDS makes use of the IP package information collected by network hardware such as switches and rooters. Although this kind of information is not so abundant as that of host-based IDS, network-based IDS has preponderance in detecting so-called "distributed" intrusions among the whole network and lighten the burden on every individual host machine.

There are generally two distinct approaches in the field of intrusion detection: misuse detection and anomaly detection [1, 10]. Misuse detection utilizes attack signatures, usually taking the form of rules, to detect intrusion. It gains a high detection rate for those well-known intrusions, but often fails to detect novel intrusions. Anomaly detection, however, tries to build up normal profiles, the patterns of normal behaviors. Any deviant from the normal profiles is considered as anomalies [21]. Because it is difficult to precisely establish the normal profiles, anomaly detection usually suffers from a higher false positive rate, the possibility that a normal behavior is mistakenly classified as an attack instance.

There have been plenty of methods in intrusion detection. A statistical method is proposed in [2], where several "metrics" are paid attention to and their statistical normal profiles are constructed. Enlightened by that, many researchers try to build statistical models of a host system from various aspects [13, 14]. Data mining is also widely studied and used in intrusion detection [8, 11]. It focuses on extracting so-called "association rules" and "frequent episodes" from voluminous data, which are a specific kind of rules to describe the network activities.

Recently, it is particularly popular to utilize the methods in machine learning and pattern recognition to detect intrusions. For unsupervised learning, SOM has engaged broad attention [3, 9] due to its excellent clustering performance and easy implementation. In [21], a hierarchy framework of using SOM is proposed, which achieves an eminent detecting result. As to supervised learning, ANN is a common tool [17,24], but SVM gets more favors in virtue of its great generalization ability [7,25].

While these existing methods can obtain a high detection rate (DR), they often suffer from a relatively high false positive rate (FPR), which wastes a great deal of manpower. Meanwhile, their computational complexities are also oppressively high, which limits their applications in practice, because an IDS would affect the regular tasks of the target systems if it emploits too much resource.

Adaboost is one of the most prevailing machine learning algorithms in recent years. Its computational complexity is generally lower than SOM, ANN and SVM in the case that the size of the data set is voluminous while the dimensionality is not too high. For this and other advantages, we employ Adaboost algorithm for our network-based IDS.

Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)
0-7695-2415-X/05 $20.00 © 2005 **IEEE**

This rest of paper is organized as follows. The framework of our IDS is proposed in section 2. In section 3, we describe how the fatal parts of our IDS work in details. The classical Adaboost algorithm is introduced and an improvement on it is proposed. The analysis about the computational complexity of our IDS can be found in section 4. In section 5, experiment results are provided. At last, we draw some conclusions in section 6.

2 Framework of our IDS

We have constructed a network-based IDS, and its framework (as shown in Figure 1) is made up four modules:

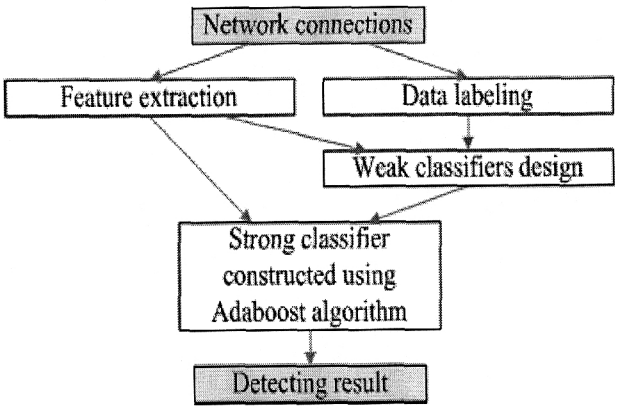

Figure 1: **Framework of our network-based IDS**

- **Feature extraction.** For every network connection, we extract three major groups of features for detecting intrusions: "Basic features of individual TCP connections", "Content features within a connection suggested by domain knowledge" and "Traffic features computed using a two-second time window" [23]. The framework for constructing these features can be found in [10].
- **Data labeling.** Because Adaboost is a supervised learning algorithm, we have to label a set of data for training. This labeled data set should contain both normal samples, labeled as "+1", and attack samples, labeled as "-1". So our algorithm is neither "misuse detection" nor "anomaly detection" mentioned above, but a kind of "hybrid detection".
- **Weak classifiers design.** Adaboost requires a group of weak classifiers designed before hand. "Weak (or basic)" means that the classifying accuracy of an individual classifier is relatively low. In section 3.1, weak classifiers used in our IDS is described.
- **Strong classifiers construction.** A strong classifier is constructed using our "improved" Adaboost algorithm. We will show the details of this procedure in section 3.2 and 3.3.

After training, a strong classifier is obtained. Then a new network connection represented by the same three groups of features can be send to the strong classifier and classified as either "normal" or "attack", shown in Figure 1 as detecting result.

3 Methodology

3.1 Weak classifiers design

A group of weak classifiers has to be prepared as inputs of Adaboost algorithm. They can be linear classifiers, ANNs or other common classifiers. In our algorithm, we select "decision stumps" as weak classifiers due to its simpleness.

For every feature f, its value range could be divided into two nonoverlapping value subsets $\mathcal{C}_p^f$ and $\mathcal{C}_n^f$, and the decision stump on f takes the form as follow:

$$h_f(\mathbf{x}) = \begin{cases} +1 & \mathbf{x}(f) \in \mathcal{C}_p^f \\ -1 & \mathbf{x}(f) \in \mathcal{C}_n^f \end{cases}$$

where $\mathbf{x}(f)$ indicates the value of $\mathbf{x}$ on feature f. The minimal total error rate criterion is used to determine the two value subsets:

$$(\mathcal{C}_p^f, \mathcal{C}_n^f) = \arg \min_{(\widetilde{\mathcal{C}}_p^f, \widetilde{\mathcal{C}}_n^f)} [\varepsilon_{\widetilde{h}_f}^+(\widetilde{\mathcal{C}}_p^f, \widetilde{\mathcal{C}}_n^f) + \varepsilon_{\widetilde{h}_f}^-(\widetilde{\mathcal{C}}_p^f, \widetilde{\mathcal{C}}_n^f)]$$

where $\varepsilon_{\widetilde{h}_f}^+$ and $\varepsilon_{\widetilde{h}_f}^-$ respectively denotes the classifying error rates of normal samples and attack samples by the decision stump $\widetilde{h}_f$.

3.2 Classical Adaboost algorithm

Adaboost is a stereotype algorithm of boosting, whose basic idea is to select and combine a group of weak classifiers to form a strong classifier [6]. The classical Adaboost algorithm is shown in Table 1, where n is the size of the training set, and $\mathcal{H} = \{h_f\}$ is a set of weak classifiers. It has been proved that the objective function:

$$\varepsilon_{tr} = \frac{1}{n} |\{i : H(\mathbf{x}_i) \neq y_i\}| \quad (1)$$

has an upper bound $\prod_t Z_t$, and in every loop, Z_t achieves its minimum $2\sqrt{\varepsilon_t(1-\varepsilon_t)}$ by choosing $\alpha_t = \frac{1}{2}\log(\frac{1-\varepsilon_t}{\varepsilon_t})$. This ensure that the training error of the strong classifier converges to zero exponentially to the number of rounds [6].

Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)
0-7695-2415-X/05 $20.00 © 2005 **IEEE**

Table 1: **Classical Adaboost algorithm**

Given: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where $y_i \in \{+1, -1\}$

Initialize weights: $D_1(i) = 1/n$ $(i = 1, \ldots, n)$

For $t = 1, \ldots, T$:

1. Choose a weak classifier h_t which minimizes the weighted error:

$$h_t = \arg\min_{h_j \in \mathcal{H}} \varepsilon_j = \sum_{i=1}^{n} D_t(i) I[y_i \neq h_j(\mathbf{x}_i)]$$

2. If $\varepsilon_t = \min_j \varepsilon_j > 1/2$, set $T = t-1$ and stop loop.
3. Choose an α_t.
4. Update the weights:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}$$

where Z_t is a normalization factor assuring D_{t+1} is a distribution.

The strong classifier is:

$$H(\mathbf{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}))$$

3.3 Improved objective function and initialized weights

The objective function (1) of the classical Adaboost algorithm is not very suitable for the problem we are facing. In intrusion detection, we have to pay more attention to FPRs, because a high FPR wastes a great deal of manpower. In our IDS, we employ an "improved objective function" to adjust the tradeoff between FPR and DR:

$$\begin{aligned} \varepsilon_{wr} = & \frac{r}{n_+}|\{i : y_i = +1, H(\mathbf{x}_i) = -1\}| \\ & + \frac{1-r}{n_-}|\{i : y_i = -1, H(\mathbf{x}_i) = +1\}| \end{aligned} \quad (2)$$

where n_+ and n_- are respectively the number of normal samples and attack samples in the training set. r is a scale factor to punish DRs.

In the theory of Adaboost, the initialized weights $D_1(i) = 1/n$ has a strong relationship with the objective function (1). So corresponding to our improved objective function, a kind of "improved initialized weights" is used:

$$D_1(i) = \begin{cases} \frac{r}{n_+} & y_i = +1 \\ \frac{1-r}{n_-} & y_i = -1 \end{cases} (i = 1, \ldots, n) \quad (3)$$

where r is the same as the r in (2). Here we can find that r also indicates the importance of the normal samples. The larger r is, the more weightily normal samples are treated at the beginning of the algorithm.

4 Computational complexity

Under simple analysis, we can easily calculate that in the training phase of the Adaboost algorithm, the computational complexity is only $O(nTM)$, where M is the number of decision stumps. As to SOM or ANN, the computational complexity of the training phase depends on the distribution of the data set, and in the worst case it is $O(n^2M^2)$, which is higher than Adaboost especially when n is large as in the data set we used. As to SVM, although there is a popular accelerating algorithm named SMO, the complexity of training is also exponential to n in general cases.

For the testing phase, the computational complexity of Adaboost is $O(n'T)$, where n' is the number of the input records. It is also lower than that of hierarchical SOM, which is $O(n'M^2)$ mentioned in [21], because T is commonly in the same quantitative level as M.

In a word, Adaboost generally possesses lower computational complexity than SOM, ANN and SVM, especially in the training phase. This property is very attractive and promising in intrusion detection, because the classifiers should be retrained in short periods in practice and fast detection helps to activate the following defending measures in due course.

5 Experiments

5.1 Intrusion Data Set

We utilize the KDD CUP 1999 data set [23] for our experiments. It was originated from MIT's Lincoln Lab and developed for IDS evaluations by DARPA [19]. Despite of several drawbacks mentioned in [18], it has served as a reliable benchmark data set for many researches on network-based intrusion detection algorithms. In this data set, each TCP/IP connection has been labeled, and 41 features had been extracted, some of which are continuous and others are categorical. So we don't have to do the task of "Feature extraction" and "Data labeling" shown in Figure 1, then we can focus on the effectiveness and efficiency of the core algorithm of our IDS framework.

There are four general types of attacks appeared in the data set: DOS (denial of service), U2R (user to root), R2L (remote to local) and PROBE. In each of the four, there are many low level types of attacks. Detailed descriptions about the four general types can be found in [15, 19]. The number of samples of various types in the training set and the test set are listed respectively in Table 2 and Table 3. "NOVEL" in

Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)
0-7695-2415-X/05 $20.00 © 2005 **IEEE**

Table 2: **Numbers of samples of various types in the training set**

Normal	Attack				Total
	DOS	U2R	R2L	PROBE	
	391458	52	1126	4107	
97278	396743				494021

Table 3: **Numbers of samples of various types in the test set**

Normal	Attack					Total
	DOS	U2R	R2L	PROBE	NOVEL	
	223298	39	5993	2377	18729	
60593	250436					311029

Table 3 represents those low level attack types not appeared in the training set.

5.2 Classical Adaboost algorithm and improved initialized weights

First, we run the classical Adaboost algorithm, whose result is shown in Table 4. Then we run it again with the improved initialized weights in the form of (3), where r varies from 0 to 1. The results are provided in Table 5. We can see that when r is not too small nor too large, that is, from 0.3 to 0.7, the results is generally better than that of the classical Adaboost algorithm. We set $T = 40$ in all of our experiments.

Table 4: **Result of classical Adaboost algorithm**

Training Set		Test Set	
FPR(%)	DR(%)	FPR(%)	DR(%)
2.766	**99.159**	**3.428**	**90.738**

Table 5: **Results with improved initialized weights (all decision stumps)**

r	Training Set		Test Set	
	FPR(%)	DR(%)	FPR(%)	DR(%)
0	99.999	100	100	100
0.1	6.325	99.396	8.874	91.823
0.2	2.766	99.159	3.428	90.738
0.3	2.724	99.106	3.365	90.711
0.4	2.712	99.078	3.335	90.690
0.5	**0.851**	**98.519**	**2.200**	**90.140**
0.6	0.402	98.432	1.677	90.016
0.7	0.143	98.411	1.431	89.914
0.8	0.142	98.364	1.408	89.903
0.9	0.0586	98.306	0.361	89.868
1	0	0.312	0.0891	0.0371

5.3 Avoid Overfitting

In our experiments, we notice that the total error rates of the $23rd$ and the $3rd$ decision stumps on the training set are respectively 1.61% and 4.92%. These too excellent performance probably means overfitting. So we exclude these two decision stumps from $\mathcal{H}$ and the results get much improved as expected, shown in Table 6.

5.4 About the selection of r

From Table 6, we finally make the selection that $r = 0.5$, for under this value, we get the best balanced between the FPR and the DR, respectively 0.665% and 90.477% on the test set. Intuitively, $r = 0.5$ means that we pay equal attention to the FPR and the DR, and the normal sample set and the attack sample set are equally emphasized at the beginning of the algorithm. If we hope to get lower FPRs, we could moderately increase r. For example, when $r = 0.7$, the FPR decreases to 0.307% on the test set, but the DR simultaneously decreases to 90.04%. This trend is consistent with the theoretical analysis very well.

Table 6: **Results with improved initialized weights (without the 23rd, 3rd stumps)**

r	Training Set		Test Set	
	FPR(%)	DR(%)	FPR(%)	DR(%)
0	99.999	100	100	100
0.1	6.928	99.188	8.971	91.908
0.2	5.504	99.166	8.212	91.825
0.3	2.754	98.956	1.787	90.875
0.4	0.856	98.855	0.682	90.531
0.5	**0.844**	**98.791**	**0.665**	**90.477**
0.6	0.822	98.791	0.668	90.475
0.7	0.523	98.445	0.307	90.040
0.8	0.0884	98.346	0.264	89.879
0.9	0.0915	97.837	0.393	89.539
1	0	0.312	0.0891	0.0371

Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)
0-7695-2415-X/05 $20.00 © 2005 **IEEE**

5.5 Comparison with some other published results

Several recently published experiment results and our results run on the same data set are listed in Table 7. We can find that ours are greatly competitive with others in terms of an especially low FPR while keeping an agreeable DR. Bagged C5 [4, 20] is the winning algorithm of the KDD99, which a bit outperforms our algorithm in terms of FRP and DR, but it is much more time-comsuming, as mentioned below.

Table 7: **Results comparison**

Methods	FPR(%)	DR(%)
Genetic Clustering [16]	0.3	79
Hierarchical SOM [21]	2.19-3.99	90.94-93.46
SVM [5]	6-10	91-98
Bagged C5 [4, 20]	0.55	91.81
RSS-DSS [22]	0.27-3.5	89.2-94.4
Improved Adaboost	**0.31-1.79**	**90.04-90.88**

5.6 Computational Time

We did our experiments on a computer of Pentium IV, 2.6GHZ CPU, 256M RAM, and the whole algorithm is implemented in MATLAB 7. The mean training time is only 73s, using all of the 494,021 training samples. This is an empirical substantiation that the computational complexity of our IDS is especially low. While in [12], the least training time of SOM and improved competitive learning neural network are respectively 1057s and 454s only using 101,000 samples for training. Bagged C5 [4, 20] outperforms our algorithm in terms of FRP and DR, but it took a bit more than a day on a machine with a two-processor ultra-sparc2 (2x300Mhz) and 512M main memory. The latest published algorithm RSS-DSS [22] needs 15 minutes to finish the whole process on a 1 GHz Pentium III laptop with 256M RAM.

From the comparison above, we can find that our IDS has obvious predominance in term of speed, which is a much preferable property in practice.

6 Conclusion

We have constructed an IDS with Adaboost, a prevailing machine learning algorithm, and described how each part of the whole system works in this paper. An improvement concerning about getting low FPRs and balancing the importance of normal samples and attack samples have been proposed. The experiment results have shown that our IDS obtains an extremely low false positive rate with a fairish detection rate. We also have demonstrated that our IDS has a noticeable advantage in computational complexity compared with some other algorithms.

References

[1] S. Chebrolu, A. Abraham, and J. P. Thomas. Feature deduction and ensemble design of intrusion detection systems. *Computers & Security*, In Press,Corrected Proof,Available online, November 2004.

[2] D. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, February 1987.

[3] M. O. Depren, M. Topallar, E. Anarim, and K. Ciliz. Network-based anomaly intrusion detection system using soms. In *Proceedings of the IEEE 12th Signal Processing and Communications Applications Conference*, pages 76–79, April 2004.

[4] C. Elkan. Results of the kdd99 classifier learning contest. *SIGKDD Explorations*, 1(2):63–64, 2000.

[5] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security*, Chapter 4. D. Barbara and S. Jajodia (editors) Kluwer, 2002.

[6] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Computer and System Sciences*, 55(1):119–139, August 1997.

[7] P. Hong, D. Zhang, and T. Wu. An intrusion detection method based on rough set and svm algorithm. In *Proceedings of International Conference on Communications, Circuits and Systems*, volume 2, pages 1127–1130, June 2004.

[8] H. Jin, J. Sun, H. Chen, and Z. Han. A fuzzy data mining based intrusion detection model: Distributed computing systems. In *Proceedings of 10th IEEE International Workshop on Future Trends*, pages 191–197, May 2004.

[9] H. G. Kayacik, A. Zincir-Heywood, and M. Heywood. On the capability of an som based intrusion detection system. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 1808–1813, July 2003.

[10] W. Lee and S. J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security*, 3(4):227 – 261, November 2000.

[11] W. Lee, S. J. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 120–132, May 1999.

[12] J. Z. Lei and A. Ghorbani. Network intrusion detection using an improved competitive learning neural network. In *Proceedings of Second Annual Conference on Communication Networks and Services Research*, volume 4, pages 190–197, May 2004.

Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)
0-7695-2415-X/05 $20.00 © 2005 **IEEE**

[13] J. Li and C. Manikopoulos. Novel statistical network model: the hyperbolic distribution. In *IEE Proceedings on Communications*, volume 151, pages 539–548, December 2004.

[14] Z. W. Li, A. Das, and S. Nandi. Utilizing statistical characteristics of n-grams for intrusion detection. In *Proceedings of International Conference on Cyberworlds*, pages 486 – 493, December 2003.

[15] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. *ACM Transactions on Information and System Security*, 34(4):579–595, October 2000.

[16] Y. G. Liu, K. F. Chen, X. F. Liao, and W. Zhang. A genetic clustering method for intrusion detection. *Pattern Recognition*, 37(5):927–942, May 2004.

[17] Y.-H. Liu, D.-X. Tian, and A.-M. Wang. Annids: intrusion detection system based on artificial neural network. In *Proceedings of International Conference on Machine Learning and Cybernetics*, volume 3, pages 1337–1342, November 2003.

[18] J. McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information, System and Security*, 3(4):262–294, November 2000.

[19] S. Mukkamala, A. H. Sung, and A. Abraham. Intrusion detection using an ensemble of intelligent paradigms. *Network and Computer Applications*, 28(2):167–182, April 2005.

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

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

[22] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. Training genetic programming on half a million patterns: An example from anomaly detection. *IEEE Transactioins on Evolutionary Computatioin*, 9(3):255–239, June 2005.

[23] S. Stolfo and et al. The third international knowledge discovery and data mining tools competition [online]. Available: http://kdd.ics.uci.edu/databases/kddCup99/kddCup99.html, 2002.

[24] C. Zhang, J. Jiang, and M. Kamel. Intrusion detection using hierarchical neural networks. *Pattern Recognition Letters*, In Press, Corrected Proof, Available online, November 2004.

[25] Z. Zhang and H. Shen. Online training of svms for real-time intrusion detection based on improved text categorization model. *Computer Communications*, In Press, Uncorrected Proof, Available online, February 2005.

Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)
0-7695-2415-X/05 $20.00 © 2005 **IEEE**