基于改进的随机森林算法的入侵检测模型*

郭山清[1,2+], 高 丛[3], 姚 建[1,2], 谢 立[1,2]

[1](南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093)

[2](南京大学 计算机科学与技术系,江苏 南京 210093)

[3](Department of Computer Science, University of Auckland, Auckland, 1020, New Zealand)

An Intrusion Detection Model Based on Improved Random Forests Algorithm

GUO Shan-Qing[1,2+], GAO Cong[3], YAO Jian[1,2], XIE Li[1,2]

[1](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

[3](Department of Computer Science, University of Auckland, Auckland, 1020, New Zealand)

+ Corresponding author: Phn: +86-25-83202540, E-mail: guosq2002@yahoo.com.cn

Received 2004-03-07; Accepted 2004-11-03

Guo SQ, Gao C. Yao J, Xie L. An intrusion detection model based on improved random forests algorithm. *Journal of Software*, 2005,16(8):1490−1498. DOI: 10.1360/jos161490

Abstract: Coupled with the explosion of number of the network-oriented applications, Intrusion Detection as an increasingly popular area is attracting more and more research efforts. Although a number of algorithms have already been presented to tackle this problem, they are unable to achieve balanced detection performance for different types of intrusion and cannot respond as quickly as expected. Employing random forests algorithm (RFA) in intrusion detection, this paper devises an improved variation −− IRFA and presents an IRFA based model for intrusion detection in information exchanged through network connections. The feasibility in balanced detection and the effectiveness of this approach are verified by experiments based on DARPA data sets.

Key words: intrusion detection; random forests algorithm; classified tree; evolutionary algorithm

摘 要: 针对现有入侵检测算法对不同类型的攻击检测的不均衡性和对攻击的响应时间较差的问题.将随机森林算法引入到入侵检测领域,构造了基于改进的随机森林算法的入侵检测模型,并把这种算法用于基于网络连接信息的数据的攻击检测和异常发现.通过对 DARPA 数据的入侵检测实验,其结果表明,基于改进的随机森林算法的入侵检测模型是可行的、高效的,对数据集 DARPA 中所包含的 4 种类型的攻击检测具有良好的均

* Supported by the Natural Foundation of Jiangsu Province of China under Grant No.BK2002073 (江苏省自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2003AA142010 (国家高技术研究发展计划(863))

GUO Shan-Qing was born in 1976. He is a Ph.D. candidate of the Nanjing University. His current research areas are network security, machine learning etc. **GAO Cong** was born in 1978. He is an MSc student of the University of Auckland (New Zealand). His current research areas are machine learning and intelligent agent. **YAO Jian** was born in 1971. He is a Ph.D. candidate of Nanjing University, and his current research areas are security management and distributed computing. **Xie Li** was born in 1942. He is a professor and doctoral supervisor at the Department of Computer Science and Technology, Nanjing University and a CCF senior member. His research areas are distributed computing and advanced operation system.

衡性.

关键词: 入侵检测;随机森林算法;分类树;进化算法
中图法分类号: TP309 文献标识码: A

1 Introduction

Intrusion detection system (IDS) is a key component of secure information systems. The goal of IDS is to identify a set of intrusions that attempt to compromise the integrity, confidentiality or availability of a resource[1]. In the context of information systems, intrusion refers to any unauthorized access, unauthorized attempt to access or damage, or malicious use of information resources.

There exist two methods for intrusion detection: Anomaly Detection and Misuse Detection[2]. Misuse detection techniques recognize the signatures of known attacks, match the observed behaviors with those known signatures and signal intrusions when there is a matching. Misuse detection techniques are efficient and accurate in detecting known intrusions but cannot detect novel intrusions whose signature patterns are unknown. Anomaly detection techniques establish a profile of the subject's normal behaviors (norm profile), compare the observed behaviors of the subject with its norm profile and signal intrusions when the subject's observed behaviors differ significantly from its norm profile. Anomaly detection techniques can detect both novel and known attacks.

To be used in IDS's, many soft computing techniques like Neural Networks (HMM), Support Vector Machines (SVM), etc. have been extensively employed and show a good classification rate to some kinds of attacks[3−6]. But, some drawbacks still exist (e.g., unbalanced detection performance for different types of intrusions, long response time, etc.).

RFA was first introduced in Refs.[7,8] and has been adopted in various fields including astronomy, microarray analysis and drug discovery, etc[9]. Due to its distinctive features, RFA achieved quite good performance in applications in these fields. These distinctive features include:

1. Learning trees can improve human readability
2. It runs efficiently on large databases
3. It can handle thousands of input variables without variable deletion
4. It can give estimates of what variables are important in the classification
5. It generates an internal unbiased estimate of the generalization error as the forest building progress
6. It can effectively estimate missing data and maintain accuracy
7. It computes proximities between pairs of cases that can be used in clustering, locating outliers, or give interesting views of the data.

For its features and wild application we import RFA into intrusion detection system. In this paper, we build an intrusion detection model based on RFA. The model not only achieves balanced detection performance for different types of intrusions, but also can meet the time requirements for on-line intrusion detections. The rest of this paper is organized as follows. A brief introduction and theoretical analysis of standard RFA is given in Section 2. Section 3 describes our improved variation of RFA−−IRFA. In Section 4 we present the experimental results of using IRFA in intrusion detection. Section 5 summaries the main contribution of this paper and discusses some issues related to the proposed algorithm.

2 Random Forests Algorithm

RFA was put forward by Breiman[7,8]. Below we give the definition of RFA, which is drawn from Ref.[7].

Definition 1 (Random Forest). A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(x,\theta_k)\}$ where $\{\theta_k\}$ are independent, identically distributed random vectors, and each tree casts a unit vote for the most popular class at input x.

What can be deduced from this definition is that RFA is a tree classification algorithm that ensembles many random trees. To classify a new object from an input vector, the algorithm puts the input vector down each of the trees in the forest. Each tree gives a classification, which means that each tree "votes" for that class. The forest chooses the classification owning the most votes. The entire algorithm includes two important phases: the growth period of each tree and the voting period.

2.1 Random forests algorithm

2.1.1 Growth of trees

Each tree grows as follows

(1) If the number of cases in the training set is N, sample N cases as random-but with replacement from the original data. This sample will be the training set for growing the tree.

(2) If there are m_{all} input variables, a number of $m_{try}<<m_{all}$ is specified such that at each node, m variables are selected at random out of the M, and the best split on this m is used to split the node. The value of m is held constant during the Random Forests' growing.

(3) Each tree is grown to the largest extent possible. There is no pruning.

Following these steps, RFA works very differently from other tree classifiers as there are two sources of randomness when each single tree is constructed. The first source of randomness is the new training set drawn randomly from the original training set. After that a tree is grown on the training set without pruned. The second one, as introduced by Amit and Geman in Ref.[10], is a set of random input variables for each split. With those two types of randomness, the creation of an ensemble of trees leads to more stable error rates, which make RFA overcome many drawbacks existed in popular tree structured algorithms like CART, C4.5 and ID3[11].

Typically, a tree classification algorithm searches through all input descriptors to select the one that gives the best split. In RFA, m_{try} different input variables are chosen randomly, and the algorithm only searches within this scope to do the best spilt. For every tree, a new set of m_{try} random input variables are selected, but the value of parameter m_{try} is defined by the user and the default value of m_{try} is set to $\sqrt{m_{all}}$, where m_{all} is the total number of available input variables.

2.1.2 Voting

As for combing the predictions of component classification trees, the most prevailing approach is plurality voting of majority voting[12] for classification tasks. There are also many other approaches for combining predictions. In RFA, the predication of new test data is done by majority vote also. New test data runs down all n_{tree} trees in the ensemble, and the classification of each data point is recorded for each tree, then using majority vote, the final classification given to each data point is the class that receives the most votes across all n_{tree} trees. A user-defined threshold can loosen this condition. As soon as the number of the votes for a certain class A is above the threshold, it can be classified as class A.

2.2 Theory of random forests algorithm

In this section, we briefly introduce the theory of RFA that explains why a good performance can be achieved. These results were derived by Breiman, and a more detailed description and the derivation of the theorems can be found in Ref.[7].

To grow the kth tree, a random vector θ_k independent of the past vectors $\theta_1,\theta_2,\ldots,\theta_{k-1}$ is generated. They all have the same distribution. The kth tree grows based on θ_k, and the result is a classifier $h(x,\theta_k)$, where x is the input vector. A large set of trees are generated, and the result is done based on majority vote of these trees.

In Ref.[7], a theoretical background of Random Forests is introduced, with proofs of two important theories. The author defined a margin function

$$\mathrm{mg}(X,Y)= av_k I(h_k(x)=y)-\max_{j\neq y} av_k I(h_k(x)=j) \tag{1}$$

where $I(\bullet)$ is the indicator function. The margin measures the extent to which the average number of voters at X, Y for the right class exceeds the average vote for any other class. The larger the margin, the more confidence in the classification. Based on this function, the generalization error is given by

$$PE^* = P_{x,y}(mg(x,y)<0) \tag{2}$$

Theorem 1. As the number of trees increases, for almost surely all sequences θ_1,.. PE^* converge to

$$P_{x,y}(p_\theta(h(x,\theta)=y)-\max_{j\neq Y} p_\theta(h(x,\theta)=j)<0)\ .$$

Theorem 1 is proved in Ref.[7] with the Strong Law of Large Number, showing that Random Forests does not overfit. This is a key feature of RFA and with more trees added, the generalization error PE^* will converge to a limited value, which shows that RFA has a better generalization on new unseen examples.

Theorem 2. An upper bound for the generalization error is given by $PE^* \le \overline{p}(1-s^2)/s^2$.

Although the bound is likely to be loose, it fulfills the same suggestive function for RFA as VC-type bounds do for other types of classifiers. It shows that the two ingredients involved in the generalization error for RFA are the strength of the individual classifiers in the forest and the correlation between them in terms of the raw margin function. In understanding the functioning of random forests, this ration will be a helpful guide—the smaller it is, the better.

3 Improved Random Forests Algorithm

We employ standard RFA in solving intrusion detection problems and find that RFA achieve acceptable classification performance but the running speed corresponding to some data sets are too slow to satisfy normal needs for effective detection. Therefore, improvements have to be made. According to Ref.[13], when the component learner is a neural network it may be better to ensemble some rather than all of the trees. Such a claim is the basis on which we devise our algorithm.

In order to shorten the response time (or improve the intrusion detection speed), some randomly selected trees can be excluded from the forest. Zhou *et al*.[13] proposed an algorithm named GASEN to build the selective ensembles. GASEN assigns a random weight to each of the available component learner and employs a genetic algorithm to evolve those weights so that they can to some extent characterize the fitness of the learners in joining the ensemble. GASEN then selects a learner whose weight is larger than a presser threshold to constitute the ensemble. References [13,14] reveal that it is better to ensemble part rather than all of the neural networks. However, the goodness of such a process has not yet been proved in theory.

Each individual in the evolving population is a weight vector $w=(w_1,w_2,\ldots,w_T)$, where w_i is the weight for the i-th random tree component learner. In order to evaluate the goodness of RF, a validation data set is utilized. Let E^V_w denote the estimated generalization error of the ensemble learner corresponding to the individual w on the validation set V. It is obvious that E^V_w can express the goodness of w in the way that the smaller E^V_w is, the better W is. So,

$f(w)=E^V_w$ can be the fitness function.

IRFA is modified in some way instead of using weight representation, which is, assigning a weight to each component tree and then selecting trees according to the evolved weights. Bit strings are used where "1" denotes a tree appearing in the ensemble while "0" denotes its absence. Such a bit representation gets rid of the need for manually setting the threshold for selecting component learners according to their evolved weights. IRFA is shown in Algorithm 1, where from original data set T, a random forest R consisting of random trees $t_1,t_2,\ldots,t_n$ is grown. Now if there are sufficient training data, a forest R^* with a better performance can be produced.

Algorithm 1. (IRFA)

Input: training set T, learner L

Output: Random Forests R^*

Notation:

T: data set

T_1: training data set

T_2: testing dataset

R: random forests

P: predication accuracy

b: evolving string

b^*: evolved string

Process:

1. Divide the original data set T into two data sets T_1, T_2 equal in number.
2. Produce the random forest R from the data set T_1 and get the accuracy of predication P
3. Generate a population of bit string b with all bits equal 1
4. Evolve the population with the least 1 included in the bit string b
5. Combine the random forests with the bit string b where "1" denotes a tree appearing in the ensemble while "0" denotes its absence
6. The fitness of a string b is measured as $f(b)=\sum_{x_i\in s_v:\left[\arg\max_{y\in Y}\sum_{b_t=1:C_t(x_i)=y}1\right]\neq y_i}1 \Big/ m$ and when the classification ratio p' of the new random forest is greater than P, then $P=p'$ where $p'=1-f(b)$
7. b^* is the evolved best bit string

$$C^*(x)=\arg\max_{y\in Y}\sum_{b_i^*=1:c_i(x)=y}1$$

In this algorithm, $f(b)=\sum_{x_i\in s_v:\left[\arg\max_{y\in Y}\sum_{b_t=1:C_t(x_i)=y}1\right]\neq y_i}1 \Big/ m$ is defined as the fitness function where S_v is a validation subset randomly sampled from the test data set T_2, and m is the size of S_v. In f(b), $\sum_{x_i\in s_v:\left[\arg\max_{y\in Y}\sum_{b_t=1:C_t(x_i)=y}1\right]\neq y_i}1$ denotes the error number and $f(b)$ equals the false positive ratio of classifier R when the random forest R produced using string b is utilized to classify the validation set S_v. Apparently the smaller $f(b)$, the better the classifier R.

4 Experiments

4.1 Intrusion data set

The data set used in our experiment originates from MIT's Lincoln Laboratory, and has been developed for IDS evaluations by DARPA. The LAN was operated like a real environment, being blasted with multiple attacks. In each TCP/IP connection, 41 various quantitative and qualitative features were extracted and form a new data set[15].

The four different categories of attack patterns are:

(a) Denial of Services attack (DOS). Examples are Apache2, Land, Mail bomb, SYN Flood, Ping of death, Process table, Smurf and Syslogd.

(b) User to Super user or Root Attacks (U2R). Examples are Eject, Ffbconfig, Fdformant, loadmodule, Perl, Ps and Xterm.

(c) Remote to User Attack (R2L). Examples are Dictionary, Ftp_write, Gest, Imap, Named, Phf, Sendmail , Xlock and Xsnoop.

(d) Probing a class of attacks in which an attack scans a network of computers to gather information or find known vulnerabilities. Examples are lpseep , Mscan, Nmap, Saint, Satan.

We performed five trials of the experiment for intrusion by normal attack, Probe attack, DOS attack, U2R attack and R2L attack, respectively. The data sets for these five sub-experiments were generated from a data set in KDD[16], which contains 492 000 data points. These five data sets were used to probe the detection ability to the five types of intrusion.

4.2 Preprocess of the data sets

Some variables in the input data set extracted from KDD are represented by plain text, thus are not compatible with our IRFA. So, some automated parsers must be used to complete the task of transferring the randomly selected raw TCP/IP dump data to tree-readable form. Besides, we also define some metric to measure the performance of the algorithm, as listed below:

Classification Precision (CP)=True active samples/Total samples;

False Positive Ratio (FPR)=False true samples/True sample;

Detective Ratio (DR)=True inactive samples/inactive samples;

Average Detection Time (ADT)=Total detection time/Total samples.

4.3 Experimental methodology

In the experiments, we firstly grew a forest consisting of 500 component trees by RFA using the original training data set, and got precision P. Then we grew another forest consisting of 354 component trees with precision p' using IRFA. Though the training time achieved by IRFA is a little bit longer than that by RFA, the intrusion response time is markedly improved by 30%. Also, the classified ratio by IRFA is better than that by the standard RFA. Detailed results are listed in Table 1 and Table 2.

Table 1 Performance of random forests for 5 classifications

Class	Testing time		Accuracy (%)	
	RFA	IRFA	RFA	IRFA
Normal	0.29	0.23	98.65	99.11
Probe	0.26	0.25	99.95	99.93
Dos	0.27	0.24	99.97	99.95
U2R	0.17	0.14	99.92	99.94
R2L	0.26	0.25	92.36	94.87

Table 2 Results of the FN and DR of the random forests

Class	DR (%)		FPR (%)	
	RFA	IRFA	RFA	IRFA
Normal	1.56	1.55	3.52	3.52
Probe	0.42	0.42	0	0.01
Dos	0.042	0.043	0.11	0.11
U2R	0.04	0.04	4.72	4.21
R2L	5.41	4.38	16.19	13.12

Elimination of insignificant or useless inputs leads to a simplification of the problems. As a result, faster and more accurate detection may be achieved[17]. So, how to rank features for every kind of attack is a crucial issue in the intrusion detection problems. In Ref.[18], Srinivas Mukkamala et al described a SVM and neural network based algorithm, which is designed to help determine the important features for every kind of attack. We directly employed this method in our IRFA without modification. Specifically, in every tree grown in the forest, we put down the Out-of-Bag (OOB) cases and count the number of votes casting for the correct class. If we randomly permute the values of the variable m in the OOB, then we put down the case and compute the changes of the voter, and the changes of the correct class can be thought as a kind of measure rules of the importance of the variables. By doing this, we succeed to compute the important feature set for every kind of attack (see Table 3).

Table 3 The important feature set to every kind of attack

Item	Important Variables
Normal	1,2,3,4,5,6,10,11,12,13,22,23,24,27,29,30,3132,33,34,36,35,37,40,41
Probe	2,3,4,5,6,23,24,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41
U2r	1,3,5,6,10,12,13,14,16,17,18,22,24,32,33,34,35,36,37,38,40
R2L	1,2,3,4,5,6,10,11,22,23,24,31,32,33,34,35,36,37,40,41
Dos	1,2,3,4,5,6,8,10,12,13,23,25,26,31,32,33,35,36,37,38,39,40,41

When we perform the experiment using these variable sets rather than the previously mentioned 41 features as the inputs of IRFA, the response time to every kind of attack is improved about 20%, and the classified ratio is slightly improved as well.

4.4 Analysis

Comparisons of intrusion detection performance among IRFA, Wenke Lee's algorithm, neural networks, SVM and the ensemble of SVMs on the KDD data subset[19,20] are shown in Table 4.

Table 4 Comparison with Wenke LEE, SVM, BP network and the ensemble of SVMS

Class	Wenke Lee (%)	SVM (%)	BP network (%)	Ensemble of SVM (%)	IRFA (%)
Probe	97	98.57	92.71	99.86	99.93
Dos	79.9.3	99.11	97.47	99.95	99.95
U2R	75.0	64	48	76	99.94
R2L	60.0	97.33	95.02	99.64	94.87

From Table 4 we can easily see that the IRFA achieved balances performance on the four attack patterns, and such performance is generally better than that by the other four paradigms. Specifically, all of the other four algorithms are unable to effectively handle the four attack patterns at the same time, and are weak to some types of intrusion. For instance, Wenke Lee to Dos and U2R (79.93% and 75.0%, respectively), SVM to U2R (64%), BP to U2R (48%), and ESVM to U2R (76%). Compared with that, the ratio by IRFA never goes below 94%. Therefore, IRFA is more efficient and applicable to applications in reality.

Compared with RFA and BP network, our IRFA improves the response time by 20%~35%, comparable with what can be expected by SVM. When we perform the experiments with important features, the classified ratio (including False Positive Ratio and Detective Ratio) is further improved, and the response time is shortened by 20%

than that of IRFA accepting the 41-feature set as inputs. IRFA maintains the response time to every network connection at a level of milliseconds, significantly better than SVM and BP Network do. The fact that the training time for IRFA is slightly longer than that of the other four paradigms does not affect its value for applications in reality.

5 Conclusions and Future Work

In this paper we present an intrusion detection model based on an improved variation of RFA, and verify its efficiency by theoretical analysis and experiments. Though for some types of intrusion our model does not outperform the three existing paradigms (SVM, BP Network and Ensemble of the SVMs) in terms of testing ability, its overall performance is significantly better than that of those three approaches. Furthermore, our model also achieves balanced performance for five popular attack patterns and an averagely shorter detection time for every attack, which can meet the time requirements for on-line intrusion detection. Our IRFA based model is an efficient and reliable mechanism for intrusion detection.

The future work focuses on 1) Applying relevant domain knowledge of the security field to the proposed model to further improve its detection ability for new types of attack, 2) Extracting rules from this model to help increase the security of the entire environment and improve its ability to resolve intrusion.

References:

[1] Denning D. Intrusion-Detection model. IEEE Trans. on Software Engineering, 1987,SE-13(2):222−232.

[2] Lee W, Stolfo SJ, Mok KW. A mining framework for building intrusion detection models. In: Proc. of the 1999 IEEE Symp. on Security and Privacy. 1999. 120−132.

[3] Mukkamala S, Janoski G, Sung AH. Intrusion detection using support vector machines and neural networks. In: Proc. of the IEEE Int'l Joint Conf. on Neural Networks. 2002. 1702−1707.

[4] Mukkamala A, Sung AH. Identifying significant features for network forensic analysis using artificial intelligence techniques. Int'l Journal on Digital Evidence, 2003,1(4):1−17.

[5] Nguyen BV. Introduction support vector machines and application to the computer security of anomaly detection. Presentation at Applied and Computational Mathematics Seminar. 2003-07.

[6] Denning DE. Protection and defense of intrusion. Presented at Conf. on National Security in the Information Age, US Air Force Academy, 1996.

[7] Breiman L. Random forests. Machine Learning, 2001,45(1):5−32.

[8] Breiman L. Manual on setting up, using, and understanding random forests V4.0. 2003. http://oz.Berkeley.edu/users/breiman/Using_random_forests_V4.0.pdf

[9] Remlinger K. Introduction and application of random forest on high thoughput screening data from drug discovery. In: Proc. of the Workshop for the SAMSI Program on Data Mining and Machine Learning. 2003.

[10] Amit, Y, Geman D. Shape quantization and recognition with randomized trees. Neural Computation, 1997,9(7),1545−1588.

[11] Quanlan JR. C 4.5: Programs for Machine Learning. Morgan Kaufmann Kaufmann, San Francisco, CA, 1993.

[12] Hansen LK, Salamon P. Neural network ensembles. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1990,12(10): 993−1001.

[13] Zhou ZH, Wu JX, Tang W. Ensembling neural networks: Many could be better than all. Artificial Intelligence, 2002,137(1,2): 239−263.

[14] Zhou ZH, Jiang Y. NeC4.5: Neural ensemble based C4.5. IEEE Trans. on Knowledge and Data Engineering, 2004,16(6):770−773.

[15] Stolfo SJ, Fan W, Lee W, Prodromidis A, Chan PK. Cost-Based modeling for fraud and intrusion detection: Results from the JAM project. In: Proc. of the 2000 DARPA Information Survivability Conf. and Exposition (DISCEX 2000). 2000.

[16] The UCI Knowledge Discovery in Databases Archive. 2005. http://kdd.ics.uci.edu

[17] John GH, Kohavi R, Peger P. Irrelevant features and the subset selection problem. In: Cohen WW, Hirsh H, eds. Machine Learning: Proc. of the Eleventh Int'l Conf. San Francisco: Morgan Kaufmann Publishers, 1994.

[18] Mukkamala S, Sung AH. Feature ranking and selection for intrusion detection using support vector machines. In: Proc. of the Int'l Conf. on Information and Knowledge Engineering. 2002. 503−509.

[19] Lee W, Stolof SJ, Mok KW. A data mining framework for building intrusion detection models. In: Proc. of the 1999 Symp. on Security and Privasy. Oakland, 1999.

[20] Mukkamala S, Sung AH, Abraham A. Intrusion detection using ensemble of soft computing paradigms. In: Proc. of the 3rd Int'l Conf. on Intelligent Systems Design and Applications, Intelligent Systems Design and Applications, Advances in Soft Computing. Berlin: Springer-Verlag, 2003. 239−248.

****