Network Intrusion Detection using Random Forests

Jiong Zhang and Mohammad Zulkernine
School of Computing
Queen's University, Kingston
Ontario, Canada K7L 3N6
{zhang, mzulker} @cs.queensu.ca

Abstract

Network Intrusion Detection Systems (NIDSs) have become an important component in network security infrastructure. Currently, many NIDSs are rule-based systems whose performances highly depend on their rule sets. Unfortunately, due to the huge volume of network traffic, coding the rules by security experts becomes difficult and time-consuming. Since data mining techniques can build intrusion detection models adaptively, data mining-based NIDSs have significant advantages over rule-based NIDSs. Therefore, we apply one of the efficient data mining algorithms called random forests for network intrusion detection. The NIDS can be employed to detect intrusions online. In this paper, we discuss the approaches for handling imbalanced intrusions, selecting features, and optimizing the parameters of random forests. We also report our experimental results over the KDD'99 datasets. The results show that the proposed approach provides better performance compared to the best results from the KDD'99 contest.

Keywords: *Intrusion detection, Data mining, Random forests, Network security.*

1. Introduction

With the tremendous growth of network-based services and sensitive information on networks, network security is getting more importance than ever. Although a wide range of security technologies such as information encryption, access control, and intrusion prevention can protect network-based systems, there are still many undetected intrusions. For example, firewalls cannot prevent internal attacks. Thus, Intrusion Detection Systems (IDSs) play a vital role in network security. Network Intrusion Detection Systems (NIDSs) detect attacks by observing various network activities, while Host-based Intrusion Detection Systems (HIDSs) detect intrusions in an individual host. An IDS usually does not affect the normal network operations of the targets.

There are two major intrusion detection techniques: misuse detection and anomaly detection. Misuse detection discovers attacks based on the patterns extracted from known intrusions. Anomaly detection identifies attacks based on the significant deviations from the established profiles of normal activities. Misuse detection has low false positive rate, but cannot detect novel attacks. Anomaly detection can detect unknown attacks, but has high false positive rate.

Currently, many NIDSs such as Snort [12] are rule-based systems, which employ misuse detection techniques and have limited extensibility for novel attacks. Their performances highly rely on the rules identified by the security experts. However, the amount of network traffic is huge, and it is very difficult to specify some intrusions using the rules. Therefore, the process of encoding rules is expensive and slow.

To overcome the limitations of the rule-based systems, a number of IDSs employ data mining techniques. Data mining is the analysis of (often large) observational data sets to find patterns or models that are both understandable and useful to the data owner [23]. Data mining can efficiently extract patterns of intrusions for misuse detection, establish profiles of normal network activities for anomaly detection, and build classifiers to detect attacks, especially for the vast amount of audit data. Data mining-based systems are more flexible and deployable. The security experts only need to label the audit data to indicate intrusions instead of hand-coding rules for intrusions.

Over the past several years, a growing number of research projects have applied data mining to intrusion detection with different algorithms [2, 8, 9]. For instance, MADAM ID [2] and ADAM [8] employ association rules algorithm. We propose an approach to use random forests algorithm in intrusion detection. Random forests is an ensemble classification and regression approach, which is unsurpassable in accuracy among current data mining algorithms [7]. Random forests algorithm has

been used extensively in different applications. For instance, it has been applied to prediction [16,19], probability estimation [17], and pattern analysis in multimedia information retrieval and bioinformatics [18]. Unfortunately, to the best of our knowledge, random forests algorithm has not been applied in automatic intrusion detection.

Accuracy is critical to develop effective NIDSs, since high false positive rate or low detection rate will make NIDSs unusable. To improve detection performance, we also propose methods to address the issues of imbalanced intrusions and feature selection in mining process as discussed below.

One of the challenges in intrusion detection systems is feature selection. Many algorithms are sensitive to the number of features. Hence, feature selection is essential for improving detection rate. Moreover, the raw data of network traffic is usually audited in tcpdump format, and the tcpdump format is not suitable for detection. IDSs must construct features from the raw data. The process of feature construction from tcpdump format data involves a lot of computation. Thus, feature selection can help reducing the computational cost for feature construction. However, in many current data mining-based IDSs, feature selection is based on domain knowledge or intuition. We use the feature selection algorithm of random forests, because the algorithm can give estimates of what features are important in the classification.

Another challenge of intrusion detection is imbalanced intrusion. Some intrusions such as Denial of Service (DoS) [10] have much more connections than others (*e.g.*, User to Root). Most of the data mining algorithms try to minimize the overall error rate. However, this leads to increasing the error rate of minority intrusions. However, in real world network environment, the minority attacks are more dangerous than the majority attacks. Therefore, we need to improve the detection performance for the minority intrusions.

The proposed approach is evaluated using the KDD'99 datasets, which were used for the third International Knowledge Discovery and Data Mining Tools Competition [11]. The contest involved building a classifier for detecting computer network intrusions from a very large database of network traffic. Our experimental results show that the detection performance is improved by our approach of using random forests algorithm.

The paper is organized as follows. In Section 2, we describe in detail the approach to mine audited data using random forests. The experiments and performance evaluations are presented in Section 3. In Section 4, we discuss the related work. Finally, we summarize the paper and outline our future research plans in Section 5.

2. Mining patterns of intrusions

In this section, we describe the methods employed in the proposed framework, and illustrate how to apply these methods to build detection patterns with the high performance for intrusion detection.

2.1. Overview of the framework

The proposed framework applies data mining techniques to build patterns for network intrusion detection. The working environment of the proposed NIDS is shown in Figure 1. There are two phases in the framework: off-line phase and on-line phase. The system builds patterns of intrusion in the off-line phase and detect intrusions in the on-line phase.

In the off-line phase, we feed training dataset into the Pattern Builder module which can build the patterns of intrusions. The module employs the feature selection algorithm, handles imbalanced intrusions, and builds the patterns by random forests with optimal parameters. After mining the patterns for intrusions, the module outputs the patterns as the input of the Detector module.

In the on-line phase, the system captures the packets from network traffic. The features for each connection are constructed by the preprocessors from the captured network traffic. Then, in the Detector module, the connections are classified as different intrusions or normal traffic using the patterns built in the off-line phase. Finally, the system raises an alert when it detects any intrusion.

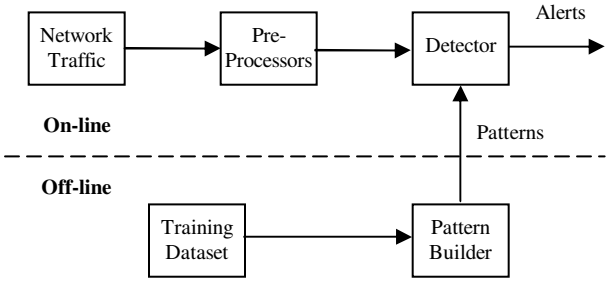

Figure 1. Working environment of the NIDS

2.2. Random forests

The random forests [7] is an ensemble of unpruned classification or regression trees. Random forest generates many classification trees. Each tree is constructed by a different bootstrap sample from the original data using a tree classification algorithm. After

the forest is formed, a new object that needs to be classified is put down each of the tree in the forest for classification. Each tree gives a vote that indicates the tree's decision about the class of the object. The forest chooses the class with the most votes for the object.

The main features of the random forests algorithm are listed as follows:

- It is unsurpassable in accuracy among the current data mining algorithms.
- It runs efficiently on large data sets with many features.
- It can give the estimates of what features are important.
- It has no nominal data problem and does not over-fit.
- It can handle unbalanced data sets.

In random forests, there is no need for cross-validation or a test set to get an unbiased estimate of the test error. Since each tree is constructed using the bootstrap sample, approximately one-third of the cases are left out of the bootstrap samples and not used in training. These cases are called out of bag (oob) cases. These oob cases are used to get a run-time unbiased estimate of the classification error as trees are added to the forest.

2.3. Optimization for random forests

The error rate of a forest depends on the correlation between any two trees and the strength of each tree in the forest. Increasing the correlation increases the error rate of the forest. The strength of a tree relies on the error rate of the tree. Increasing the strength decreases the error rate of the forest. When the forest is growing, random features are selected at random out of the all features in the training data. The best split on these random features is used to split the node. The number of random features (*Mtry*) is held constant. Reducing (Increasing) *Mtry* reduces (increases) both the correlation and the strength. The number of features employed in splitting each node for each tree is the primary tuning parameter (*Mtry*). To improve the performance of random forests, this parameter should be optimized.

We use training data to find the optimal value of the parameter *Mtry*. The minimum error rate corresponds to the optimal value. Therefore, we use the different value of *Mtry* to build the forest, and evaluate the error rate of the forest. Then, we select the value corresponding to the minimum error rate to build the pattern.

There are two ways to evaluate the error rate. One is to split the dataset into training part and test part. We can employ the training part to build the forest, and then use the test part to calculate the error rate. Another way is to use the oob error estimate. Because random forests algorithm calculates the oob error during the training phase, we do not need to split the training data. We choose the oob error estimate, since it is more effective by learning from the whole training dataset.

2.4. Imbalanced intrusions

Intrusions are imbalanced. In other words, some intrusions produce much more connections than others. Random forests algorithm tries to minimize the overall error rate, by lowering the error rate on majority classes (*e.g.*, majority intrusions) while increasing the error rate of minority classes (*e.g.*, minority intrusions) [7]. However, the damage cost due to the minority intrusions is much higher than the cost of the majority intrusions. Thus, for imbalanced intrusions, we need to improve the detection rate of the minority intrusions while maintaining a reasonable overall detection rate.

There are two solutions to deal with the imbalanced intrusions problem. One is to set different weights for different intrusions. The minority intrusions are assigned higher weights. Although the overall error rate goes up, the error rate of the minority intrusions will be reduced. Random forests algorithm supports this method by changing the weight parameters. The other method is to use sampling techniques: over-sampling the minority intrusions and down-sampling the majority intrusions. Since the network traffic is huge, down-sampling the majority intrusions (*e.g.*, normal traffic and Denial of Service) can speed up building the patterns significantly by reducing the size of the datasets. Over-sampling minority intrusions (*e.g.*, User to Root and Remote to Local) can raise their weights to decrease their error rate. Therefore, we combine over-sampling and down-sampling in our NIDS to solve the imbalanced intrusions problem instead of the first solution.

2.5. Feature selection

The raw audit data of network traffic is not suitable for intrusion detection. Hence, feature construction is needed to extract a set of features which can detect intrusions effectively. Usually, the construction is based on each connection. Each connectionless packet (*e.g.*, UDP packet) is also treated as a connection. There are three types of features for network connection records used in NIDSs [2]:

- **Intrinsic features**. Intrinsic features describe the basic information of connections, such as the

duration, service, source and destination host, port, and flag.

- **Traffic features**. These features are based on statistics, such as number of connections to the same host as the current connection within a time window.
- **Content features**. These features are constructed from the payload of traffic packets instead of packet headers, such as number of failed logins, whether logged in as root, and number of accesses to control files.

Feature selection is one of the critical steps in building NIDSs. The number of intrinsic features is fixed, since the number depends on the information of packet header. However, traffic features and content features can be constructed using different methods. Hundreds of traffic and content features can be designed, while only some of them are essential for separating intrusions from normal traffic. Unessential features not only increase computational cost, but also increase the error rate, especially for some algorithms that are sensitive to the number of features. "Deciding upon the right set of features is difficult and time consuming" [4]. Currently, features are designed by domain knowledge experts. Thus, we need an approach that can automate the feature selection. We employ variable importance calculated by random forests. The features with higher value of variable importance have more effect on classification. Therefore, we choose the features with the higher value of variable importance in the NIDS.

3. Experiments and results

In this section, we summarize our experimental results to build patterns for intrusion detection over the KDD'99 datasets. We first describe the experiments of using sampling techniques for imbalanced intrusions and random forests to select features. Then, we present the experiments on parameter optimization for random forests. Finally, we evaluate our approach and compare our results with the best result of the KDD'99 contest.

3.1. Dataset and preprocessing

Under the sponsorship of Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), MIT Lincoln Laboratory has collected and distributed the datasets for the evaluation of computer network intrusion detection systems [10, 24]. DARPA dataset is the most popular dataset used to test and evaluate a large number of IDSs. The KDD'99 dataset is a subset of DARPA dataset prepared by Sal Stofo and Wenke Lee [6]. The data was preprocessed by extracting 41 features from the tcpdump data in the 1998 DARPA datasets. The KDD'99 dataset can be used without further time-consuming preprocessing and different IDSs can compare with each other by working on the same dataset. Therefore, we carry out our experiments on the KDD'99 dataset and compare our results with the best result of the KDD'99 contest.

The KDD'99 dataset includes the full training set, the 10% training set, and the test set. The full training set has 4,898,431 connections. The 10% training set has 494,020 connections. The attacks in the dataset fall into four categories [11]: DoS (Denial of Service), R2L (unauthorized access from a remote machine), U2R (unauthorized access to root privileges), and probing. The 10% training set contains all the minority classes (U2R and R2L) of the full training set and part of the majority classes. It is just like down-sampling the majority classes (Normal, DoS and Probing). Hence, we just use the 10% training dataset in our experiments. The task of the KDD'99 contest was to build a classifier capable of distinguishing between four kinds of intrusions and normal traffic numbered as one of five classes (see Table 1).

Table 1. Numbering of the attack categories [6]

0	Normal
1	Probe
2	DoS
3	U2R
4	R2L

3.2. Performance comparison on balanced and imbalanced dataset

The original dataset (the 10% training set) is imbalanced (*e.g.*, DoS has 391,458 connections but U2R only has 52 connections). To make a balanced training set, we down-sample the Normal and DoS classes by randomly selecting 10% of connections belonging to Normal and DoS from the original dataset. We also over-sample U2R and R2L by replicating their connections. The balanced training set with 60,620 connections is much smaller than the original one.

The first experiment is to compare the performance of detection between the patterns built on the original training set and the balanced training set with sampling. The experiment is carried out by using the default values of the parameters for random forests in WEKA (Waikato

Environment for Knowledge Analysis) [21]: 66% samples as training data, 34% samples as test data, 10 trees in the forest, and 6 random features to split the nodes. The main objective of the experiment is to compare the performance differences between the balanced and the original datasets, but not to compare the effect of the parameters. As a result, for the sake of convenience, we just use the default values of the parameters for both datasets. Table 2 lists overall error rate for classification, time to build pattern, true positive rate for all classes, and false positive rate for the classes. As shown in the table, the sampling techniques can improve the performance, especially for the detection rate (true positive rate) of the minority classes (Class 3 and Class 4) and can reduce the time to build the patterns dramatically.

Table 2. Performance on the balanced dataset compared to the original dataset

Performance	Original dataset	Balanced dataset
Overall error rate	1.92%	0.05%
Time to build pattern	1975 seconds	65 seconds
True positive rate(Class 0)	0.948	0.999
True positive rate(Class 1)	0.989	0.994
True positive rate(Class 2)	1	1
True positive rate(Class 3)	0.862	1
True positive rate(Class 4)	0.83	1
False positive rate(Class 0)	0.011	0
False positive rate(Class 1)	0	0
False positive rate(Class 2)	0	0
False positive rate(Class 3)	0	0
False positive rate(Class 4)	0.01	0

3.3. Selection of important features

The second experiment is to select the most important features. There are 41 features in the KDD'99 dataset numbered from 1 to 41. They cover all three types of features in NIDSs: intrinsic features, traffic features, and content features. We employ the feature selection algorithm supported by random forests to calculate the value of variable importance. To estimate the importance of the variable *m*, the number of votes for the correct class is counted using the oob cases in every tree. Then, the number of the correct votes is counted again after randomly permuting the values of variable *m* in the oob cases. The average of the margin between these two numbers over all the trees in the forest is the raw importance score for variable *m*. The raw score is divided by its standard error to get a z-score, and the value of variable importance is the negative z-score for variable *m*. Figure 2 plots the values of variable importance for all five categories, sorted in decreasing order. The figure shows the variable importance values of the last 3 features (Feature 7, 20, and 21) are much less than other values. Therefore, we select the rest of the 38 most important features to build the patterns for intrusion detection.

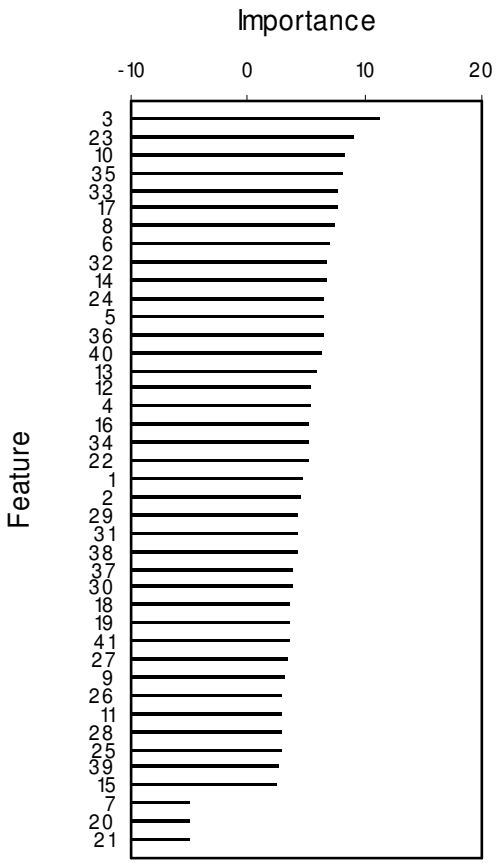

Figure 2. Variable importance of the features

The feature 3 (service type such as http, telnet, and ftp) is the most important feature to detect intrusions. It means that the intrusions are sensitive to service type.

The feature 7 (land) is used to indicate if connection is from/to the same host. According to the domain knowledge, it is the most discriminating feature for land attacks. However, land attack belongs to DoS and has much less connections than other types of DoS. After down-sampling DoS attacks, the land attacks are almost excluded from the balanced dataset. Therefore, the feature 7 is not important to improve the detection rate of DoS attacks.

Feature 20 (# of outbound commands in a FTP session) and 21 (hot login to indicate if it is a hot login) do not show any variation for intrusion detection in the training set.

The above analysis suggests that the feature selection can help choose features to detect intrusions without special domain knowledge. However, the method has high dependence on training sets.

3.4. Parameter optimization for random forests

To improve the detection rate, we optimize the number of the random features (*Mtry*). We build the forest with different *Mtry* (5, 10, 15, 20, 25, 30, 35, and 38) over the balanced training set, then plot the oob error rate and the time to build the pattern corresponding to different *Mtry*. As Figure 3 shows, the oob error rate reaches the minimum when *Mtry* is 15, 25, or 30. Besides, increasing *Mtry* increases the time to build the pattern. Thus, we choose 15 as the optimal value, which reaches the minimum of the oob error rate and costs the least time among these three values.

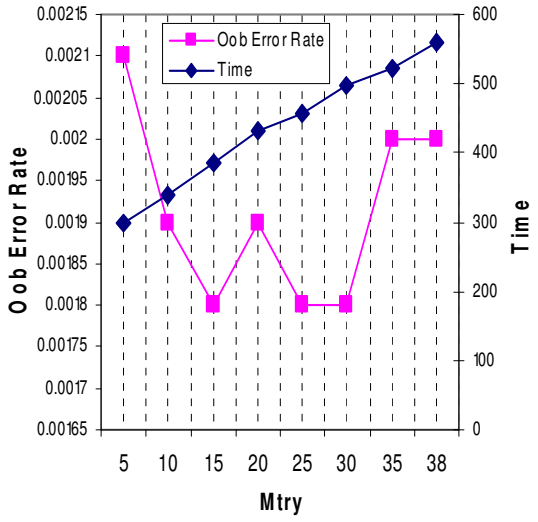

Figure 3. Performance with different values for parameter *Mtry* of random forests

3.5. Evaluation and discussion

Different misclassifications have different levels of consequences. For example, misclassifying R2L as Normal is more dangerous than misclassifying DoS as Normal. We use the cost matrix as Table 3 published in KDD'99 [6] to measure the damage of misclassification. M_{ij} denotes the number of samples in Class i misclassified as Class j, and C_{ij} indicates the corresponding cost in the cost matrix. Let N be the total number of the samples. The cost that indicates the average damage of misclassification for each connection is computed as:

$$\text{cost} = \sum M_{ij} \times C_{ij} / N$$

Similar to KDD'99 contest, we evaluate our approach with the same test dataset which contains 311,029 examples. We carry out our experiment with 50 trees and 15 random features (optimized in the previous experiments). First, we build patterns on the balanced training set using all the 41 features. Then, we build patterns using the 38 most important features. The evaluation results of the patterns are reported in Table 4 along with the best result of the KDD'99 contest. The overall error rate is the ratio of the misclassified connections to the total connections in the test set.

Table 3. Cost matrix [6]

	Normal	Probe	DoS	U2R	R2L
Normal	0	1	2	2	2
Probe	1	0	2	2	2
DoS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

Table 4. Performance comparison on the KDD'99 dataset

Experiments	Overall error rate	Cost	Time (Seconds)
Best KDD Result	7.29%	0.2331	Not provided
Experiment without feature selection	7.19%	0.2306	491
Experiment with feature selection	7.07%	0.2282	423

There were 24 participants in total in the KDD'99 contest [6]. The best result of the KDD'99 contest is listed in Table 4 achieved by an ensemble of decision trees. The experimental results show that our approach provides lower overall error rate and cost compared to the best KDD'99 result even without feature selection. The results also show that the overall error rate, cost, and time to build patterns is reduced by selecting the most important features. Thus, feature selection can improve the performance of intrusion detection.

3.6. Implementation

In our experiments, we use the random forests algorithm implemented in the WEKA (Waikato Environment for Knowledge Analysis) [21]. WEKA is a Java package which contains machine learning algorithms for data mining tasks. However, WEKA does not implement the variable importance function in the random forests algorithm. Therefore, we also use the FORTRAN 77 program [22] developed by Leo Breiman and Adele Cutler to calculate the variable importance in feature selection.

4. Related work

In recent years, many data mining-based research work have been proposed for intrusion detection. MADAM ID (Mining Audit Data for Automated Models for Intrusion Detection) [2] is one of the best known data mining projects in intrusion detection. It is an off-line IDS to produce anomaly and misuse intrusion models. Association rules and frequent episodes are applied in MADAM ID to replace hand-coded intrusion patterns and profiles with the learned rules. Association rules algorithm is used to find intra-audit record patterns, and frequent episodes algorithm is used to find inter-audit record patterns. The feature selection and construction in MADAM ID heavily relies on the expert knowledge of intrusion detection. There are five kinds of features based on their importance: flag, the axis feature, the reference feature, the rest of the essential features, and the rest of the features. The procedure to parse a frequent episode depends on the order of importance of the features.

ADAM (Audit Data Analysis and Mining) [8] is the second most widely known and well published project in the field. It is an on-line network-based IDS. ADAM can detect known attacks as well as unknown attacks. Association rules and classification, two data mining techniques, are used in ADAM. ADAM builds the profile of normal behavior from attack-free training data and represents the profile as a set of association rules. At run-time, ADAM detects suspicious connections according to the profile. Then, ADAM uses the built classifier to detect known attacks from the suspicious connections. The undetected connections are regarded as unknown attacks.

IDDM (Intrusion Detection using Data Mining Techniques) [9] is a real-time NIDS for misuse and anomaly detection. It applied association rules, meta-rules, and characteristic rules. IDDM employs data mining to produce description of network data and uses this information for deviation analysis. When detecting large deviations between the descriptions, the system can produce appropriate alerts. IDDM uses association rules algorithm to generate observations about network traffic for further processing instead of building the detection patterns directly. The higher level processing such as meta-learning builds the patterns from the observations.

In contrast to the previously proposed data mining based IDSs, we employ random forests for intrusion detection. Random forests algorithm is more accurate and efficient on large dataset like network traffic. We also use the data mining techniques to select features and handle imbalanced intrusion problem.

Other data mining algorithms are also applied to intrusion detection. For example, decision tree and fuzzy association rules are employed in intrusion detection [13,15]. Neural network is used to improve the performance of intrusion detection [14]. Support Vector Machine (SVM) is used for unsupervised anomaly detection [20].

5. Conclusion and future work

In this paper, we employ random forests algorithm in NIDSs to improve detection performance. To increase the detection rate of the minority intrusions, we build the balanced dataset by over-sampling the minority classes and down-sampling the majority classes. Random forests can build patterns more efficiently over the balanced dataset, which is much smaller than the original one. The experiments have shown that the approach can reduce the time to build patterns dramatically and increase the detection rate of the minority intrusions.

Instead of selecting features based on the domain knowledge, we select features automatically according to their variable importance calculated by random forests. By the feature selection algorithm, deciding upon the right set of features has become easy and automated. Although the approach reduces the dependency on the domain knowledge in feature selection, it increases the dependency on training sets.

From the experiments on various values of random features, we obtain the optimal value to improve the performance of random forests. The evaluation on the

KDD'99 test set shows the performance provided by our approach is better than the best result of the KDD'99 contest (reduction of the overall error rate and the cost of misclassification).

In our current approach, we only use random forests in misuse detection. Random forests algorithm also supports the outlier detection which can be used to detect novel attacks. Outliers are cases that have significant deviation from the main body of the data. Random forests algorithm uses proximity to detect outliers. Further study may be carried out to see whether the outlier detection provided by random forests can be implemented effectively in NIDSs. We also plan to investigate other data mining methods for anomaly detection.

The architecture of a NIDS is important, and we need to develop an extensible architecture with high performance. We plan to employ both misuse and anomaly detection. Misuse detection can reduce false positive rate and anomaly detection can detect novel intrusions. We have started to work on multiple classifier architecture whose overall performance will be higher than the performance of each classifier built by random forests.

6. Acknowledgements

This research work is partially funded by Bell Canada through Bell University Laboratories (BUL). The authors would like to thank David Skillicorn for his helpful comments on an initial draft of this paper.

7. References

[1] Wenke Lee, Sal Stolfo, and Kui Mok, "Adaptive Intrusion Detection: A Data Mining Approach", *Artificial Intelligence Review*, Kluwer Academic Publishers, 14(6):533-567, December 2000.

[2] Wenke Lee and Salvatore J. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems", *ACM Transactions on Information and System Security (TISSEC)*, Volume 3, Issue 4, November 2000.

[3] Wenke Lee, Sal Stolfo, Phil Chan, Eleazar Eskin, Wei Fan, Matt Miller, Shlomo Hershkop, and Junxin Zhang, "Real Time Data Mining-based Intrusion Detection", *The 2001 DARPA Information Survivability Conference and Exposition (DISCEX II)*, Anaheim, CA, June 2001.

[4] W. Lee and S. J. Stolfo, "Data Mining Approaches for Intrusion Detection", *the 7th USENIX Security Symposium*, San Antonio, TX, January 1998.

[5] Yongguang Zhang, Wenke Lee, and Yi-An Huang, "Intrusion Detection Techniques for Mobile Wireless Networks", *Wireless Networks*, Volume 9, Issue 5, September 2003.

[6] Charles Elkan, "Results of the KDD'99 Classifier Learning", *SIGKDD Explorations* 1(2): 63-64, 2000.

[7] L. Breiman, "Random Forests", *Machine Learning* 45(1):5–32, 2001.

[8] Daniel Barbarra, Julia Couto, Sushil Jajodia, Leonard Popyack, and Ningning Wu, "ADAM: Detecting Intrusions by Data Mining", *Proceedings of the 2001 IEEE, Workshop on Information Assurance and Security T1A3 1100 United States Military Academy*, West Point, NY, June 2001.

[9] Tamas Abraham, "IDDM: Intrusion Detection Using Data Mining Techniques", DSTO Electronics and Surveillance Research Laboratory, Salisbury, Australia, May 2001.

[10] M. Mahoney and P. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection", *Proceeding of Recent Advances in Intrusion Detection (RAID)-2003*, Pittsburgh, USA, September 2003.

[11] KDD'99 datasets, The UCI KDD Archive, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, Irvine, CA, USA, 1999.

[12] Snort, Network Intrusion Detection System, http://www.snort.org.

[13] Sara Matzner Chris Sinclair, and Lyn Pierce, "An Application of Machine Learning to Network Intrusion Detection", *Proceedings of the 15th Annual Computer Security Applications Conference*, pp. 371–377, Phoenix, AZ, USA, 1999.

[14] Richard P. Lippmann and Robert K. Cunningham, "Improving Intrusion Detection Performance Using Keyword Selection and Neural Networks", *Computer Networks*, 34:597–603, 2000.

[15] Jianxiong Luo and Susan M. Bridges, "Mining Fuzzy Association Rules and Fuzzy Frequency Episodes for Intrusion Detection", *International Journal of Intelligent Systems*, Vol. 15, No. 8, pp.687-704, 2000.

[16] Lan Guo, Yan Ma, Bojan Cukic, and Harshinder Singh, "Robust Prediction of Fault-Proneness by Random Forests", *Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE'04)*, pp. 417-428, Brittany, France, November 2004.

[17] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng, "Probability Estimates for Multi-class Classification by Pairwise Coupling", *The Journal of Machine Learning Research*, Volume 5, December 2004.

[18] Yimin Wu, *High-dimensional Pattern Analysis in Multimedia Information Retrieval and Bioinformatics*, Doctoral Thesis, State University of New York, January 2004.

[19] Bogdan E. Popescu, and Jerome H. Friedman, *Ensemble Learning for Prediction*, Doctoral Thesis, Stanford University, January 2004.

[20] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Salvatore Stolfo. "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data." *Applications of Data Mining in Computer Security*, 2002.

[21] WEKA software, Machine Learning, http://www.cs.waikato.ac.nz/ml/weka/, The University of Waikato, Hamilton, New Zealand.

[22] Leo Breiman and Adele Cutler, Random forests, http://stat-www.berkeley.edu/users/breiman/RandomForests/cc_home.htm, University of California, Berkeley, CA, USA.

[23] David J. Hand, Heikki Mannila, and Padhraic Smyth, *Principles of Data Mining*, The MIT Press, August, 2001.

[24] MIT Lincoln Laboratory, DARPA Intrusion Detection Evaluation, http://www.ll.mit.edu/IST/ideval/, MA, USA.