A Lightweight Network Anomaly Detection Technique

Jinoh Kim*†, Wucherl Yoo†, Alex Sim†, Sang C. Suh*, Ikkyun Kim‡
*Texas A&M University, Commerce, TX 75428 **
Lawrence Berkeley National Laboratory, Berkeley, CA 94720 †
ETRI, Daejeon, Korea ‡
Email: jinoh.kim@tamuc.edu, wyoo@lbl.gov, asim@lbl.gov, sang.suh@tamuc.edu, ikkim21@etri.re.kr

***Abstract*—While the network anomaly detection is essential in network operations and management, it becomes further challenging to perform the first line of detection against the exponentially increasing volume of network traffic. In this work, we develop a technique for the first line of online anomaly detection with two important considerations: (i) availability of traffic attributes during the monitoring time, and (ii) computational scalability for streaming data. The presented learning technique is lightweight and highly scalable with the beauty of approximation based on the grid partitioning of the given dimensional space. With the public traffic traces of KDD Cup 1999 and NSL-KDD, we show that our technique yields 98.5% and 83% of detection accuracy, respectively, only with a couple of readily available traffic attributes that can be obtained without the help of post-processing. The results are at least comparable with the classical learning methods including decision tree and random forest, with approximately two orders of magnitude faster learning performance.**

1. Introduction

The world is highly interconnected with the greater use of network-based applications. Accordingly the risk of intrusions and cyber-attacks has been intensified over the past decades. For instance, the distributed denial of service attack (DDoS) is prevalent with the rise of botnets and attack tools easily accessible. In addition to DDoS attacks, there are several other types of numerous anomalous activities that attempt to gain unauthorized access to protected resources, discover network services and resources, and destabilize the network on the whole. To keep the network safe and stable, identifying anomalous activities is a critical problem in local and ISP networks [3], [5], [12].

A traditional approach to network anomaly detection is based on the examination of the content of packets with a pre-constructed signature table containing common textual patterns [14]. While highly accurate, the overhead of the payload inspection is very heavy, and it is often not feasible to catch up the line rate even with the expensive specialized hardware. Other concerns with this approach is the increasing use of encryption and the growing privacy regulations. An alternative approach relies on machine learning techniques, such as classification and clustering [7], [9], [10], [15], with the statistical information to detect anomalies. Since it does not assume to examine the payload section, the cost of operation is relatively cheap, and it also relaxes the concerns of privacy and encryption. With these benefits, we take the learning-based approach to develop a technique for the first line of network anomaly detection in this work.

There are two considerations in our design. First, the first line of detection is performed in an online computation manner. Thus, the referenced traffic variables considered in this phase should be readily available without the need of post-processing. For instance, the KDD Cup data set [2] that has been widely employed for the anomaly detection study, consists of 41 attributes categorized into three groups: the "basic" group including the connection-related information, "content" with the additional information to look for suspicious activities, and "traffic" with the aggregated information within a two-second time window. The attributes in the basic group are readily available in the traffic collection time, whereas the variables in the other two groups can be obtained through post-processing with the domain-specific knowlege. In this circumstance, the basic group attributes are the only ones that can be considered for online detection.

The second consideration in our design is scalability since it is a key challenge in network monitoring with the exponential increase of the network traffic [1], [4]. It is much critical to online processing to keep up the line rate with the limited computing resources. For this reason, only a few selected variables are often accounted for data streaming computation [8]. However, the main focus of the past work for anomaly detection was more on maximizing classification accuracy with as many variables as possible [9], [15]. Unlike this, we give a greater priority to scalability to promote streaming processing and assume only a small set of the readily available attributes for detection. Additionally, we consider the complexity of the learning cost in our design to facilitate future updates of the trained model [6].

In this paper, we present a new technique for the first line of network anomaly detection designed with the above considerations. The proposed method is lightweight with a grid-based approximation borrowed from [11]. It recursively partitions the dimensional space in question until it can label the subspaces, which are then retrieved to distinguish anomalous connections from the normal. Only with a couple of traffic variables ("src_bytes" and "dst_bytes") readily available for online processing, we will show that the proposed tech-

Algorithm 1 Proposed learning method

```
1: procedure PARTITION(block b, level l)
2:     if l = L then
3:         label(b') ← 'Not Sure'
4:         return
5:     B' ← partition b into 2^l subblocks by cutting each axis by half
6:     for all b' ∈ B' do:
7:         if all population in b' have the same label c then
8:             label(b') ← c
9:         else
10:            Partition(b', l+1)
11: procedure MAIN
12:    S : D-dimensional space
13:    L: the max level to stop
14:    Place data points on S
15:    level ← 1
16:    Partition(S, level)
```

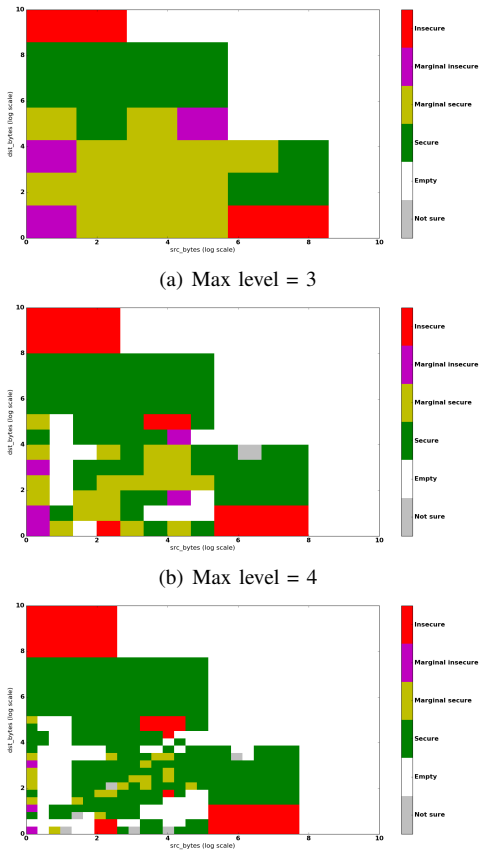
nique yields 98.5% and 83% of detection accuracy with the traditional KDD Cup data set [2] and NSL-KDD (a modified version of the KDD Cup data set) [13], respectively. The observed results are at least comparable with the classical learning methods including decision tree and random forest, with the significantly smaller learning cost (two orders of magnitude faster).

2. The Proposed Learning Method

In this section, we present our proposed technique based on the approximation using the grid-structured partitioning. The main idea of the proposed technique is to partition the D-dimensional space (S) in a top-down manner, and Algorithm 1 illustrates the details. In the algorithm, we first place all the data points on S. Each axis from the first to the D-th dimension is divided into two equal lengths at each level, from one to L (= *MAX_LEVEL*). At each level l, the max number of partitions is thus equal to 2^l. For each sub-block after partitioning, we test whether all the data points in the given block have the same label (c); If true, we label the block as c; otherwise, the procedure $Partition()$ is invoked in a recursive way until the current level l reaches L. In case of D=2, the cell is represented as a rectangle, while it is a cuboid if D=3, and so forth.

The complexity of data placement is proportional to the number of data points N (i.e., $O(N)$). The partitioning complexity is proportional to the maximum number of cells, i.e., $O(2^{D \cdot L})$. Hence, the overall complexity is $\max(O(N), O(2^{D \cdot L}))$. If we assume a large number of data points (for scalable analysis), N would be a dominant factor, and the algorithm complexity for learning would be converged to $O(N)$.

Algorithm 1 labels the pure cells only, which means all the inhabitants in the cell belong to the identical class. For flexibility, we can relax this condition with a procedure named $marginal\_check()$, by which we label a non-pure cell if the fraction of the majority is greater than (1-m), where m is a user-defined marginal value. We assume that the marginal check takes place after completing partitioning. The complexity of this additional check is also proportional to the number of cells. The procedure is straightforward and omitted to illustrate.



(a) Max level = 3

(b) Max level = 4

(c) Max level = 5

Figure 1. Partitioning with different max levels

Figure 1 represents the partitioning results with different max levels from L=3 to L=5 with margin m=0.3 in a 2-dimensional space. The data set used here is the first quarter of 'kddcup.'data_10_percent_corrected" in the KDD Cup data set. From Figure 1, the max number of cells is 64 (=8 $\times$ 8) with L=3, while there can be (32 $\times$ 32) cells at the max when L=5. The marginal value indicates that if the majority population is greater than 70% in the given block space, it is labeled as the class of the majority. As shown from the figure, there are six classes: "Secure", "Marginal Secure", "Marginal Insecure", "Insecure", "Empty", and "Not Sure". Here, "Secure" and "Insecure" refer to pure blocks with a single class for 100% of the inhabitants. The two types of marginal labels can be applied to non-pure blocks based on m: If the fraction of the majority population is greater than m, it is labelled as one of "Marginal Secure" and "Marginal Insecure". The partition is labelled as "Not Sure" in case of no presence of the majority. "Empty" is the unknown space with the given data to learn.

Testing a connection is straightforward using the map constructed in the learning phase: If the connection resides in any of the secure cells, it is considered as a

normal connection; if it is in any of the insecure cells, the connection is an anomalous one; otherwise, it is impossible to determine and classified into "unknown". The testing complexity is $O(log(L))$ like typical tree-based algorithms. With a simple optimization, the complexity for testing can be down to $O(1)$ with the storage requirement of the max number of cells.

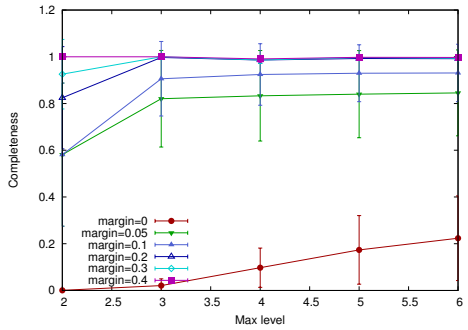
3. Evaluation

3.1. Description of data sets

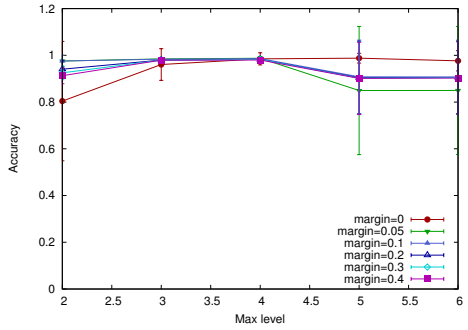
We evaluate the proposed technique with KDD Cup 10% data ("kddcup.'data_10_percent_corrected") [2] and NSL-KDD (a refined version of the KDD Cup data by substantially removing redundant information) [13]. We refer to the data sets as KDD10 and NSL-KDD, respectively, throughout the paper. The number of records in KDD10 is 494K, each of which contains the associated information with the connection. The NSL-KDD data set in our experiment consists of four files, two for training and the other two for testing, and the default training and testing files ("KDDTrain+" and "KDDTest+") consist of 126K and 22.5K connections, respectively.

A data instance in the KDD data sets is a record of a single connection, a sequence of packets with the same source and destination IP addresses and TCP port numbers, including the following three groups of features: (i) TCP connection information (i.e., 9 features including the duration of the connection, the number of data bytes from source to destination and vice versa), (ii) content information obtained using the domain knowledge (i.e., 13 features including the number of failed login attempts and the number of root accesses), and (iii) traffic information obtained in every two-second time window (i.e., 9 features including the number of connections to the same host as the current connection in the past two seconds). The individual record also contains a label of either normal or a specific kind of attack, categorized into denial of service ("DOS"), unauthorized access from a remote host ("R2L"), unauthorized access to root functions ("'U2R"), and surveillance and other probing for vulnerabilities ("Probe").

Out of the provided 41 features from the data sets, we employ the attributes in the first basic group only. Again, the features within the non-basic groups can be obtained through an additional post-processing and not readily available during the traffic collection time. We mainly considered two variables in the basic connection group: "src_bytes" and "dst_bytes": the former is the number of bytes from the source to the destination IP addresses, and the latter is the number of bytes from the destination to source hosts. Among the five continuous attributes (duration, src_bytes, dst_bytes, wrong_segment and urgent) in the basic group, two variables of wrong_segments and urgent are supportive (non-primary) attributes and mostly zeros, and hence these cannot work as discriminators. The variable, duration, has also a lot of zeros due to the sampling unit of seconds (rather than miliseconds). For this reason,



(a) Completeness



(b) Accuracy

Figure 2. Evaluation results with two connection attributes of (src_bytes, dst_bytes) with the KDD data set

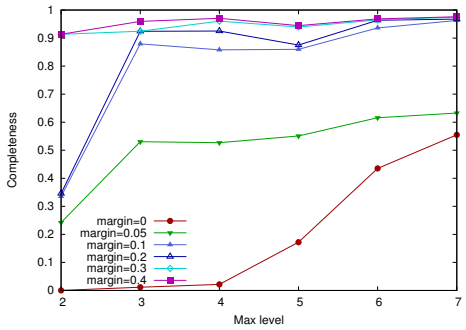
we chose the above two attributes as the primary discriminators in our experiment.

We normalized the variables to relax the skewness. For instance, the summary of src_bytes is: 0 (min), 45 (1st quartile), 520 (median), 3,026 (mean), 1,032 (3rd quartile), and 693,400,000 (max), showing a highly right-skewed distribution and the mean is three times greater than the third quartile. We normalized the data with log function and the summary after the process is: 0.00 (min), 1.65 (1st quartile), 2.72 (median), 2.16 (mean), 3.01 (3rd quartile), and 8.84 (max). Applying the log function is reasonable in the following sense: a ten-byte difference between two connections with src_bytes=10 and src_bytes=20 would be significant, but the difference of src_bytes=1,000,000 and src_bytes=1,000,010 are not much significant.

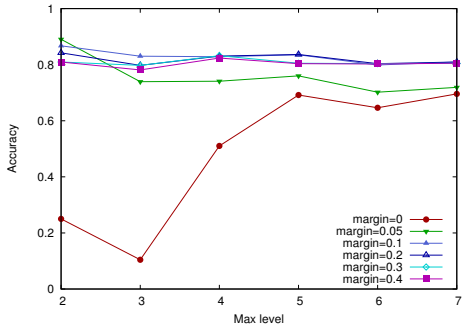
To evaluate performance, we employ two measures: "Completeness" is defined as the fraction of the connections in question that are classified either normal or anomalous; "accuracy" is defined as the fraction of the connections correctly identified out of all the connections classified either normal or anomalous.

3.2. Experimental results

We firstly present the result of the sensitive study by examining the impact of the parameters (MAX_LEVEL and margin) with both of the data sets. Then we perform a comparison study with decision tree and random forest using the data set of NSL-KDD.



(a) Completeness



(b) Accuracy

Figure 3. Evaluation results with two connection attributes of (src_bytes, dst_bytes) with the NSL-KDD data set

Figure 2 shows the impact of the parameters in different settings with $m = \{0, 0.05, 0.1, 0.2, 0.3, 0.4\}$ (for margin) and $L = \{2, 3, 4, 5, 6\}$ (for level) with the data set of KDD10. In this experiment, we split the data file into 10 sub files in a disjoint manner (i.e, 49K records per each file after splitting) and executed five times with a pair of two randomly-chosen sub-files (F_1 and F_2 and $F_1 \neq F_2$). The plots in the figure show the average with the standard deviation. As expected, m is a dominant factor for completeness as shown in Figure 2(a); we observed less than 30% of completeness with $m = 0$. With $m \geq 0.2$, we see very high completeness approaching to 100% with $L \geq 3$. Figure 2(b) shows accuracy with the same set of parameters. Overall, we observed better accuracy and completeness with tiny standard deviations ($\sigma < 0.015$) when $L = 4$ and $0.2 \leq m \leq 0.4$. We initially expected better performance with a greater L value, but it showed a high degree of variation when $L \geq 5$, perhaps due to over-fitting.

Figure 3 shows the results with a pair of KDDTrain+ and KDDTest+ for training and testing in NSL-KDD. Note that the difficulty level for classifying NSL-KDD is much greater than the original KDD data with the significant reduction of redundant records. For example, a recent study in [9] reported 82% of the binary classification accuracy with a complex model based on Naive Bayes, KNN, and LDA (Linear Discriminant Analysis) for dimension reduction with the entire features. Our technique produces 82.4% of accuracy with completeness=97.1%, when L=4 and m=0.4 only with the two attributes in consideration. From the figure, setting either $m = 0.3$ or $m = 0.4$ is a safe choice to maintain a high degree of completeness (over 96%) with the improved accuracy. With respect to the level, we observed that $L = 4$ is still a good choice. Table 1 compares the performance with the decision tree and random forest. Our technique works at least comparable to the popular learning techniques consistently, with 96.1% of completeness on average.

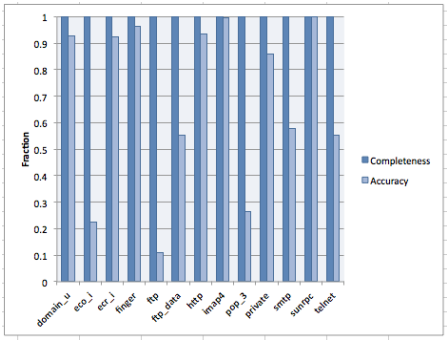


Figure 4. Completeness and accuracy by service

TABLE 1. EVALUATION RESULT WITH NSL-KDD: THE PROPOSED TECHNIQUE IS WITH D=2, L=4, m=0.4, COMPLETENESS $> 95\%$.

Run	Proposed	DT	RF
KDDTrain+/KDDTest+	82.4%	74.9%	81.1%
KDDTrain+/KDDTest-21	66.6%	66.3%	64.7%
KDDTrain_20%/ KDDTest+	82.4%	76.9%	81.9%
KDDTrain_20%/ KDDTest-21	66.6%	64.5%	62.9%

From our experiment, we observed very poor performance for some services. We next examine this more in detail by adding an extra dimension. The grid space is now 3D with the original two variables plus the new additional dimension for those services. This implies that the learning takes place independently for each service. Figure 4 shows the result by services. We include the services that have more than 100 connections in the figure. Half of the services including "http" show high accuracy over 90%, while some services such as "ftp", "ecr_o", and "pop3" show very poor accuracy less than 30%, downgrading the performance considerably. Examining this further would be an interesting avenue for the future investigation to optimize the performance.

To discuss the learning complexity with the measured data, we measured the learning time for three algorithms (decision tree, random forest and the proposed technique) on a dedicated cluster node with Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz (12 cores) and 64GB memory. For learning from "KDDtrain+", it takes 226 msec for our technique with the configuration of MAX_LEVEL=4 and margin=0.4 which is the same setting used for the performance evaluation presented in Table 1. We observed that 17.1 sec for decision tree and 13.7 sec for random forest, which are roughly two orders of magnitude slower than our technique.

4. Related work

A large body of work investigated network intrusion/anomaly detection using machine learning techniques. Various techniques were compared in [7], including both supervised and unsupervised learning with the KDDCup 1999 data. To reduce the bias in the data set, the authors conducted preprocessing that includes sampling and normalization. In their experiments, the authors observed that C4.5 (a variant of the decision tree algorithm) works the best with 95% true positive rate at 1% false positive rate, followed by MLP (Multi-Layer Perceptron) and SVM (Support Vector Machine).

Recent studies such as [9], [15] employed NSL-KDD for the evaluation. The work in [9] established a complex model with Naive Bayes, KNN, and LDA (Linear Discriminant Analysis) for dimension reduction, and reported 82% of the binary classification accuracy with the default NSL-KDD training and testing pair. The result in the study [15] with a Bayesian classifier that aggregates a selected set of Bayesian network classifiers is promising with over 95% of accuracy by utilizing multiple classifiers. The focus of our work is basically different from the past work. First, our interest is in the first line of detection with a couple of attributes readily available. The existing techniques largely considered the entire attributes with a need of an extra operation of feature selection. Again, many of the attributes in the KDD Cup data set can be available by post-processing with the specific domain knowledge. Second, the computational complexity is very expensive for a complex model with the techniques above, while our technique is lightweight and scalable based on an approximation model.

5. Conclusion

With the increasing traffic volume, scalability is one of the primary concerns for the first line of detection, and the computational complexity should be manageable to succeed. This paper presented a new method designed for practical, on-line anomaly detection using a grid partitioning approximation. We evaluated the proposed technique and showed the detection accuracy and completeness as well as computational complexity. With only two connection-related variables ("src_bytes" and "dst_bytes") readily available during the traffic monitoring time, our technique performed at least comparable with the classical learning methods including decision tree and random forest, yielding the accuracy of 98.5% with the KDD data and 83% with NSL-KDD. In addition, the measured learning time for our method is significantly low and approximately two orders of magnitude faster than decision tree and random forest.

6. Acknowledgment

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Visiting Faculty Program (VFP), by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.B0101-15-1293).

References

[1] Cisco white paper: Cisco vni forecast and methodology, 2015-2020, http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf.

[2] KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[3] P. Bodík and G. Bronevetsky, editors. *2012 Workshop on Managing Systems Automatically and Dynamically, MAD'12, Hollywood, CA, October 7*. USENIX Association, 2012.

[4] K. Cho, K. Fukuda, H. Esaki, and A. Kato. Observing slow crustal movement in residential user traffic. In *Proceedings of the 2008 ACM Conference on Emerging Network Experiment and Technology, CoNEXT 2008, Madrid, Spain, December 9-12, 2008*, page 12, 2008.

[5] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda. Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the 6th International Conference*, Co-NEXT '10, 2010.

[6] F. Jiang, Y. Sui, and C. Cao. An incremental decision tree algorithm based on rough sets and its application in intrusion detection. *Artif. Intell. Rev.*, 40(4):517–530, 2013.

[7] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck. Learning intrusion detection: Supervised or unsupervised? In *Image Analysis and Processing - ICIAP 2005, 13th International Conference, Cagliari, Italy, September 6-8*, pages 50–57, 2005.

[8] Y. Liu, L. Zhang, and Y. Guan. A distributed data streaming algorithm for network-wide traffic anomaly detection. *SIGMETRICS Perform. Eval. Rev.*, 37(2):81–82, Oct. 2009.

[9] H. H. Pajouh, G. Dastghaibyfard, and S. Hashemi. Two-tier network anomaly detection model: a machine learning approach. *Journal of Intelligent Information Systems*, pages 1–14, 2015.

[10] E. E. Papalexakis, A. Beutel, and P. Steenkiste. Network anomaly detection using co-clustering. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, ASONAM '12, pages 403–410, 2012.

[11] E. Schikuta. Grid-clustering: A fast hierarchical clustering method for very large data sets. Technical report, 1993.

[12] S. Shin and G. Gu. Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks. In *Proceedings of the 2012 20th IEEE International Conference on Network Protocols (ICNP)*, ICNP '12, pages 1–6, 2012.

[13] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, CISDA'09, 2009.

[14] Y. Wang, Y. Xiang, W. Zhou, and S. Yu. Generating regular expression signatures for network traffic classification in trusted network management. *J. Netw. Comput. Appl.*, 35(3):992–1000, May 2012.

[15] L. Xiao, Y. Chen, and C. K. Chang. Bayesian model averaging of bayesian network classifiers for intrusion detection. In *IEEE 38th Annual Computer Software and Applications Conference, COMPSAC Workshops 2014, Vasteras, Sweden*, 2014.