Network traffic anomaly detection using machine learning approaches

Conference Paper · April 2012

DOI: 10.1109/NOMS.2012.6211951

CITATIONS
9

READS
1,873

2 authors:

Kriangkrai Limthong
Bangkok University
8 PUBLICATIONS **62** CITATIONS
SEE PROFILE

Thidarat Tawsook
Bangkok University
1 PUBLICATION **9** CITATIONS
SEE PROFILE

Network Traffic Anomaly Detection using Machine Learning Approaches

Kriangkrai Limthong* and Thidarat Tawsook†
*Graduate University for Advanced Studies (Sokendai), Tokyo 101-8430, Japan
†Computer Engineering Department, Bangkok University, Pathumthani 12120, Thailand
krngkr@nii.ac.jp, thidarat.t@bu.ac.th

***Abstract*—One of the biggest challenges for both network administrators and researchers is detecting anomalies in network traffic. If they had a tool that could accurately and expeditiously detect these anomalies, they would prevent many of the serious problems caused by them. We conducted experiments in order to study the relationship between interval-based features of network traffic and several types of network anomalies by using two famous machine learning algorithms: the naïve Bayes and *k*-nearest neighbor. Our findings will help researchers and network administrators to select effective interval-based features for each particular type of anomaly, and to choose a proper machine learning algorithm for their own network system.**

***Index Terms*—anomaly detection, time interval, network traffic analysis, machine learning, naïve Bayes, nearest neighbor**

I. Introduction

Anomalies in network traffic are major symptoms of computer security problems and network congestion. Network administrators and researchers have been trying to find a method that can accurately and expeditiously perceive anomalies in network traffic. Generally, we can classify anomaly detection methods into two major groups [1]: signature-based methods and statistical-based methods.

The signature-based methods monitor and compare network packets or connections with predetermined patterns known as signatures. This technique is a simple and efficient processing of the audit data. Although the false positive rate of these techniques can also be low, comparing packets or connections with large sets of signatures is a time consuming task and has limited predictive capabilities. The signature-based methods cannot detect novel anomalies that would not be defined in the signatures, and thus administrators frequently have to update the system signatures.

The statistical-based methods, however, have the ability to learn behavior of network traffic and the possibility of detecting novel anomalies. The machine learning approach is one of the statistical-based methods that has high capabilities to automatically learn to recognize complex patterns and make intelligent decisions based on data [2]. There are two basic types of machine learning techniques: unsupervised algorithms and supervised algorithms.

The unsupervised algorithms take a set of unlabeled data as its input and attempt to find anomalies based on the assumption that the large proportion of data is normal network traffic [3]. However, it is not true in many cases, such as in denial of service (DoS) attacks or when there are accidents, outages, or misconfigurations. On the other hand, the supervised algorithms can cover and detect a wide range of network anomalies that the unsupervised algorithms cannot. The major assumption of supervised algorithms is that the anomalous network traffic is statistically different from normal network traffic. There have been many studies on supervised anomaly detection [4], such as the Bayesian network-based one [5], *k*-nearest neighbor-based one [6], and support vector machine-based one [7]. Nevertheless, a comparison of the performances of these algorithms has not yet been conducted.

978-1-4673-0269-2/12/$31.00 © 2012 IEEE

Almost all of the previous studies used packet-based or connection-based features that have a scalability issue when the number of packets or connections increases. For example, if we have network traffic consisting of 10 packets for 10 seconds long, the processing time on the packet-based features will be 10 units just like with the interval-based features. By contrast, if the network traffic consisting of 1,000 packets for 10 seconds long, the processing time on the packet-based feature will be 1,000 units but the interval-based feature still remains 10 units. For this reason, we are looking for alternate features called interval-based features that could answer the scalability problem of the packet-based or connection-based features.

The contributions of this work are to study and compare nine different interval-based features by using two famous machine learning algorithms, namely naïve Bayes and *k*-nearest neighbor.

II. Materials and Methods

The naïve Bayes algorithm is a simple probabilistic learning algorithm that is based on applying the Bayes' theorem [8], which has been applied to many domains such as document classification and face recognition. The *k*-nearest neighbor is a method for classifying objects based on the closest training examples in the feature space [9]; it is one of the simplest learning algorithms. Both of the algorithms have different strengths, weaknesses, and especially dissimilar time consumption during the training and testing phase.

A. Data Sets

We acquired three-month data traces of an anomaly-free network from an edge router of the Internet service center at Kasetsart University, Thailand. This center is for college

TABLE I
CHARACTERISTICS OF SELECTED ANOMALIES

Sources	#SrcAddr	#DstAddr	#SrcPort	#DstPort	#Packet	Packet Size Min:Avg:Max (Bytes)	Occurrence (Seconds)	#AvgPacket per Second	%Anomaly
back									
Week 2 Fri	1	1	1,013	1	43,724	60:1,292.31:1,514	651	67.16	0.75
Week 3 Wed	1	1	999	1	43,535	60:1,297.29:1,514	1,064	40.92	1.23
ipsweep									
Week 3 Wed	1	2,816	1	104	5,657	60:60.26:118	132	42.86	0.15
Week 6 Thurs	5	1,779	2	105	5,279	60:67.75:118	4,575	1.15	5.30
neptune									
Week 5 Thurs	2	1	26,547	1,024	205,457	60:60:60	3,143	65.37	3.64
Week 6 Thurs	2	1	48,932	1,024	460,780	60:60:118	6,376	72.27	7.38
Week 7 Fri	2	1	25,749	1,024	205,600	60:60:60	3,126	65.77	3.62
portsweep									
Week 5 Tues	1	1	1	1,024	1,040	60:60:60	1,024	1.02	1.19
Week 5 Thurs	1	1	1	1,015	1,031	60:60:60	1,015	1.02	1.17
Week 6 Thurs	2	2	2	1,024	1,608	60:60:60	1,029	1.56	1.19
smurf									
Week 5 Mon	7,428	1	1	1	1,931,272	14:1,066:1,066	1,868	1,033.87	2.16
Week 5 Thurs	7,428	1	1	1	1,932,325	14:1,066:1,066	1,916	1,008.52	2.22
Week 6 Thurs	7,428	1	1	1	1,498,073	1,066:1,066:1,066	1,747	857.51	2.02

students, educators, and researchers so that they can ascertain advantageous information for their studies from the Internet. There are about 1,300 users per day, and the service time is between 8:30 and 24:00 on every weekday. The users cannot change or install any software from the computer clients, and the administrators provide appropriate software for all ordinary users. Moreover, the administrators regularly update the virus signatures of the anti-virus software installed on all of the clients. At the end of every day, all the clients automatically reverse their operating system and all the software back to the initial state, so we can guarantee that all of them are clean and anomaly-free.

We selected 39 days of clean data traces to train all the classifiers in the training phase and other 16 days to combine them with several types of anomalies. The selected anomalies are from the Lincoln Laboratory at the Massachusetts Institute of Technology [10], [11]. These anomalies were provided for researchers who would like to evaluate and compare the efficiency of their own anomaly detection method.

We selected five different types of anomalies that have the characteristics listed in Table I. The main criteria we took into account are the number of source and destination address, the number of source and destination port, and the number of average packet per second. The ***back*** attack is a denial of service attack against the Apache web server through port 80, where a client requests a URL containing many backslashes. The ***ipsweep*** attack is a surveillance sweep performing either a port sweep or ping on multiple IP addresses. The ***neptune*** attack is a SYN flood denial of service attack on one or more destination ports. The ***portsweep*** attack is a surveillance sweep through many ports to determine which services are supported on a single host. The ***smurf*** attack is an amplified attack using an ICMP echo reply flood.

There are two reasons why we selected the network data traces from different sources. First, although the data traces from MIT consist of both normal and anomaly network traffic, we do need real and clean network traffic to train the classifiers because making an effective decision on all the classifiers depends on the training data. The second reason is the selected anomalies are test bed data that anyone can use it for evaluating detection methods on their own network traffic.

TABLE II
INTERVAL-BASED EVALUATION

Test Result	Actual Status	
	Anomaly	Normal
Anomaly	True Positive (TP)	False Positive (FP)
Normal	False Negative (FN)	True Negative (TN)

B. Evaluation

The measure of accuracy that we use for evaluating is F-measure [12], which takes into consideration both the precision and recall [13] of the test to compute the score. We use the precision, recall, and F-measure on a per-interval basis. All the measures can be calculated based on the following four parameters: the true positive (TP), which is the number of anomalous intervals correctly detected, the false positive (FP), which is the number of normal intervals wrongly detected as anomalous intervals, the false negative (FN), which is the number of anomalous intervals not detected, and the true negative (TN), which is the number of normal intervals correctly detected. All of the parameters are defined in Table II. From these parameters, the precision, recall, and F-measure are derived by using Eqs. (1)-(3), respectively:

$$\textit{precision} = \frac{TP}{TP + FP}, \tag{1}$$

$$\textit{recall} = \frac{TP}{TP + FN}, \tag{2}$$

$$\textit{F-measure} = 2 \times \frac{\textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}}. \tag{3}$$

In Eq. (1), the precision or positive predictive value is the percentage of detected intervals that are actually anomalies. In Eq. (2), the recall or sensitivity value is the percentage of the actual anomalous intervals that are detected. Equation (3) shows the F-measure value, which is the harmonic mean of the precision and recall. We used the F-measure value as a single measure of the classifier performance because it represents the performance of anomaly detection even better than an accuracy value or receiver operating characteristic (ROC).

TABLE III
INTERVAL-BASED NETWORK TRAFFIC FEATURES

f#	Features	Description
f1	Packet	Number of packets
f2	Byte	Sum of packet size
f3	Flow	Number of flows
f4	SrcAddr	Number of source addresses
f5	DstAddr	Number of destination addresses
f6	SrcPort	Number of source ports
f7	DstPort	Number of destination ports
f8	ΔAddr	$\lvert$SrcAddr $-$ DstAddr$\rvert$
f9	ΔPort	$\lvert$SrcPort $-$ DstPort$\rvert$

III. PRELIMINARY RESULTS

Due to the characteristics of selected anomalies, we focused on the nine interval-based features of the network traffic listed in Table III. The results include the outcome from the naïve Bayes and *k*-nearest neighbor classifications, which depict the comparison of the F-measure values across all nine features.

A. Experiment 1: Naïve Bayes Classification

In the first experiment, we varied the discriminant value and time interval value in the naïve Bayes learning algorithm to find the best F-measure value. First, we used the testing data of the ***back*** attacks by using the Packet feature (f1). Figure 1 shows the F-measure values on the ***back*** attacks by using the Packet feature (f1). Then, we located the best F-measure values from this figure. Second, we switched the feature from the Packet feature (f1) to the Byte feature (f2) and so on. After that, we changed the testing data from the ***back*** attacks to the ***ipsweep*** attacks. We went through the same process as for the ***back*** attacks and repeated it for all types of anomalies. Next, we compared the precision, recall, and F-measure values on all the features for each type of anomaly as shown in Fig. 2. The x-axis indicates the features (f1-f9) that are listed in Table III. The results show the effective features for a particular type of anomaly using the naïve Bayes algorithm.

B. Experiment 2: k-Nearest Neighbor Classification

In this experiment, we set $k = 3$ and then varied the distance value and time interval value in *k*-nearest neighbor algorithm to find the highest F-measure value. If the number of training examples measured from a testing example with the distance value is equal or greater than 3, we classify the testing example as normal. Like in the prior experiment, we first focused on the testing data of the ***back*** attacks by using the Packet feature (f1). Then, we spotted the highest F-measure

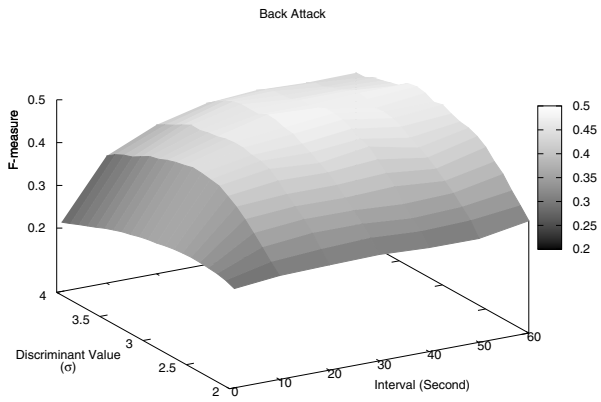

Fig. 1. F-measure of Packet Feature (f1) with Naïve Bayes

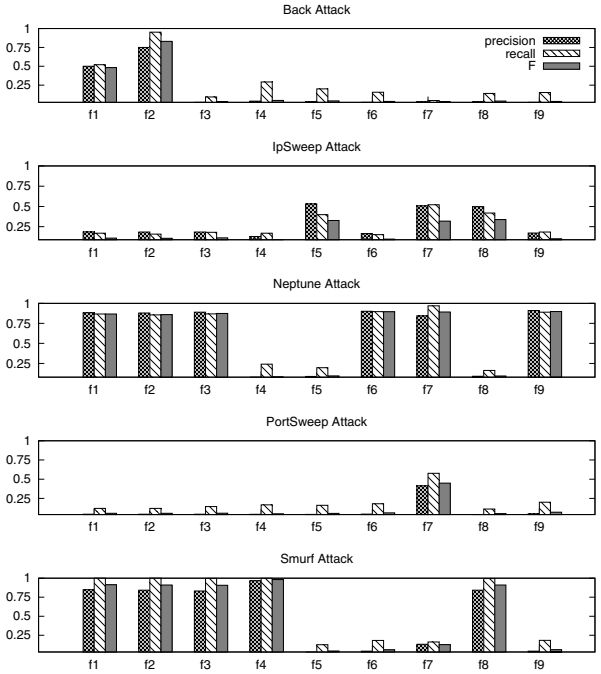

Fig. 2. Feature Comparison of Naïve Bayes Algorithm

value for the ***back*** attacks by using the Packet feature (f1) from Fig. 3. Next, we moved on different features until all nine features were covered. After that we changed the testing data to different types of anomalies just like we did with the naïve Bayes algorithm. Figure 4 shows a comparison of the precision, recall, and F-measure values for all the features of each type of anomaly. The main objective of this experiment was to compare the results from the *k*-nearest neighbor with those from the naïve Bayes algorithm.

IV. DISCUSSION

We found from preliminary results that both experiments on the two algorithms produced virtually the same results. The effective features for the ***back*** attack were the Byte feature (f2)

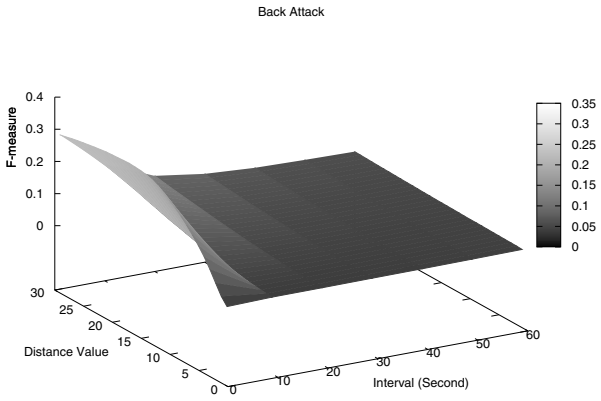

Fig. 3. F-measure of Packet Feature (f1) with *k*-Nearest Neighbor

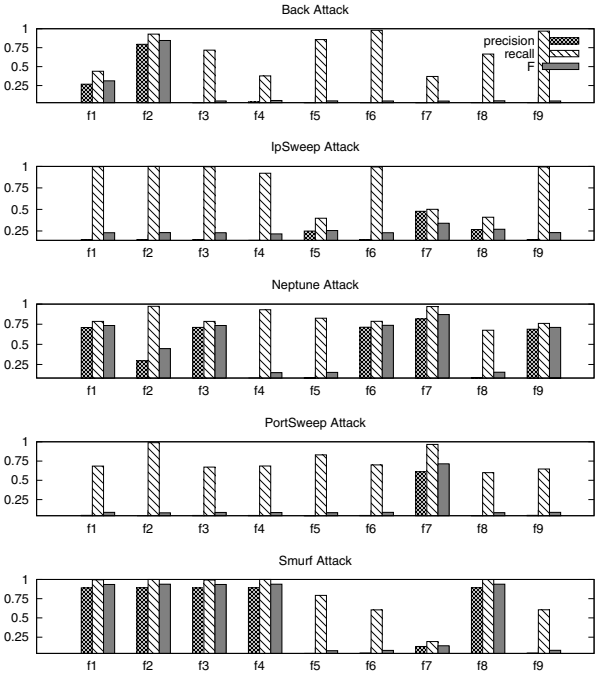

Fig. 4. Feature Comparison of *k*-Nearest Neighbor Algorithm

and Packet feature (f1), for both of the algorithms, respectively. The feasible features for the ***ipsweep*** attack were the DstAddr feature (f5), DstPort feature (f7), and the ΔAddr feature (f8). All of the features can be used for the ***neptune*** attack except the SrcAddr feature (f4), DstAddr feature (f5), and ΔAddr feature (f8). For the ***portsweep*** attack, the only feature that produces the highest F-measure value was the DstPort feature (f7). The SrcAddr feature (f4) and the ΔAddr feature (f8) can be used for the ***smurf*** attack as well as the Packet feature (f1), Byte feature (f2), and Flow feature (f3). Surprisingly, we found that almost all the features of the *k*-nearest neighbor showed higher recall values than the naïve Bayes one for the same types of anomalies.

V. Conclusion

We conducted experiments using machine learning algorithms to answer the questions as to which interval-based features of network traffic are feasible enough to detect anomalies, and searched for a technique that might possibly improve the accuracy of detecting anomalies. We selected nine interval-based features of network traffic, five types of test bed anomalies, and two machine learning algorithms to study and evaluate the accuracy of each feature. The preliminary results revealed the more practical features for each of the anomaly types, although these are only from the naïve Bayes and *k*-nearest neighbor algorithms. Our next steps are performing the experiment on the support vector machine and applying a signal processing technique, the wavelet transform, in the part of the feature extraction to improve the accuracy of anomaly detection.

Acknowledgments

We gratefully acknowledge the funding from the Faculty Members Development Scholarship Program of Bangkok University, Thailand.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, July 2009.

[2] C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," in *Computer Security Applications Conference, 1999. (ACSAC '99) Proceedings. 15th Annual*, 1999, pp. 371 –377.

[3] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*, ser. ACSC '05. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2005, pp. 333–342.

[4] P. Laskov, P. Dssel, C. Schfer, and K. Rieck, "Learning intrusion detection: Supervised or unsupervised?" in *Image Analysis and Processing ICIAP 2005*, ser. Lecture Notes in Computer Science, F. Roli and S. Vitulano, Eds. Springer Berlin / Heidelberg, 2005, vol. 3617, pp. 50–57.

[5] D. Barbará, J. Couto, S. Jajodia, and N. Wu, "Adam: a testbed for exploring the use of data mining in intrusion detection," *SIGMOD Rec.*, vol. 30, pp. 15–24, December 2001.

[6] L. (vivian) Kuang, "Dnids: A dependable network intrusion detection system using the csi-knn algorithm," 2007.

[7] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *The VLDB Journal*, vol. 16, pp. 507–521, October 2007.

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*, 2nd ed. Wiley-Interscience, Nov. 01.

[9] G. Shakhnarovich, T. Darrell, and P. Indyk, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. The MIT Press, 2006.

[10] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, "Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation," vol. 2, 2000, pp. 12 –26 vol.2.

[11] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579 – 595, 2000, recent Advances in Intrusion Detection Systems.

[12] C. J. V. Rijsbergen, *Information Retrieval*. Newton, MA, USA: Butterworth-Heinemann, 1979.

[13] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA: ACM, 2006, pp. 233–240.