3-2 Data Mining over Network Streams

BAN Tao and KADOBAYASHI Youki

Network monitoring and analyzing systems (NMASes) have been playing an active roll for preventing cyber attacks and providing practical and proactive defense against emerging threads over the Internet. This paper provides an overview of the TraceBack Network ANalyzer (TBNAN), which uses a suite of data mining algorithms to address different aspects of cyber security and facilitate network management. The various components of TBNAN such as the statisitical classification engines, anomaly detectors, and data clustering modules, could help to illustrate the status of the monitored network as well as detect different types of attacks and intrusions against the network. Our analysis shows that the functionalities implemented by TBNAN are complementary to those of traditional signature based systems, implying that both of them can be combined to enlarge the toolbox of a network administrator for efficient cyber threat and network incident countermeasure.

Keywords
Cyber security, Network monitoring and analysis system, TraceBack Network Analyzer, Traffic classification

1 Introduction

As a result of our increasing dependence on computer network commmunictions and cyber infrastructure, cyber security becomes extremely important for a wide variety of practical domains, such as Internet service providing, banking industry and e-commerce, and privacy preserving communications. The combination of emerging vulnerabilities, and threats has made cyber security an actual barrier for the further development of modern cyberspace mechanization. Conventional security solution systems, e.g., anti-virus software, SPAM email filters, etc., usually rely on extensive human effort to identify new threats, extract particular characteristics, and update the system for prevention from new emerging threats[1]. Fortunately, the research of computational intelligence (CI) has clarified the broad utility of data mining and machine learning for dynamic decision making so that much of the labor-intensive work for counter-attacking cybersecurity problems could be saved or even made more efficient by applying advanced data mining and machine learning algorithms[1]-[3].

Internet traffic monitoring and analysis has been a handy tool to fight against cyber security issues that impact the network[4]. A typical network monitoring and analyzing system (NMAS) is equipped with a function that detects the anomalous status or performance drop in the monitored computer network in an intellectual manner and sends system warnings to the network administrator. Hence, an NMAS could contribute to protection from threats adversely affecting the network and minimization of risk and loss caused by malware or abuse of important Internet resources. NMASes often find useful applications in: (1) network server load monitoring and management, (2) intrusion sensing against external intrusions, (3) IP traceback for locating attackers using IP address spoofing, and (4) system monitoring for tracking virus propagation,

malware activities, and botnet motions.

The TraceBack Network ANalyzer (TBNAN) presented in this paper has been developed as a versatile NMAS for high-speed network environment and complex protocols. One of the key design principles of TBNAN is its adaptability and versatility. TBNAN is expected to be applicable at different access points to the network for different purposes. Below is some examplary senarios that TBNAN could find its use. When deployed at an Internet backbone access point such as a university gateway, it could help to detection large-scale networks attacks or anomaly status associated with network congestion; when applied to the gateway of a local network, it could help to identify P2P file sharing clients to faciliate capacity planning; when deployed on a cloud server that is hosting multiple virtual machine clients, it could help to monitor the application behavior for proactive detection of malware contagion; when installed on a PC, it could help to detect system misconfiguration or identify misterious threads induced by malware. The key factor to enforce such adaptablity is that the monitoring and analysis engines must be lightweight so that little burden is imposed on the monitored network even for an access point with extreme high traffic throughput. On the one hand, the lightweightness relies on the flow generator that supports tunable sampling rate. On the other hand, careful system performance tuning, i.e., on accuracy and system response time, against sampling rate is also one of the key factors.

In this paper we will introduce the basic design principles of TBNAN as well as some typical senarios that it is applied for network management and threat countermeasure. The rest of this paper is organized as follows. Chapter **2** gives a general review of the TBNAN system. Chapter **3** presents detail introduction of the intelligent analysis engines embedded in TBNAN with case studies. Chapter **4** concludes the paper.

2 Overview of TBNAN

As depicted in Fig. 1, the overall system structure of TBNAN is comprised of three

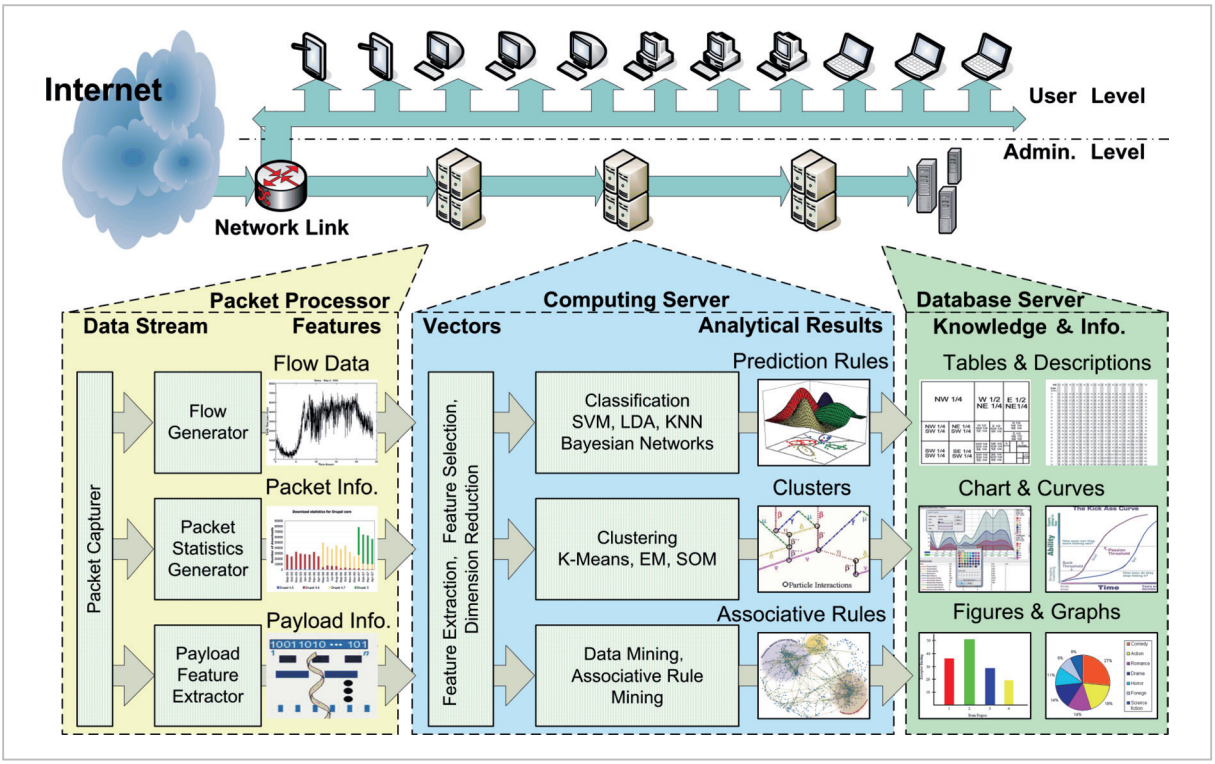


Fig.1 Overall framework of the TBNAN system

servers, i.e., a packet processor, a computation server, a database and a web server. In the following, we give a brief introuction on the three component servers.

2.1 Packet processor

The Packet Processor (PP) captures traffic that passes by an access point, extracts statistical quantities from the traffic, and then forwards the data to the computation server. To protect user privacy, PP does not inspect the payload information (depending on the situation, limited bytes of payload information could be cached for further inspection). Therefore, it circumvents reverse engineering of protocols with encrypted payload and meets the performance requirement of real-time monitoring in a next generation broadband network.

The spatial and temporal statistical quantities that PP extracts from the packet headers (i.e., data link layer, network layer, and transport layer) are termed discriminators hereinafter. Basically, discriminators are extracted on the basis of network flows, where a flow refers to a sequence of packets from a source computer to a destination. A pratical definition of flow makes use of source IP, destination IP, protocol, source port, and destination port, as this 5-tuple could uniquely identify a TCP/IP communication session within a certain time period. The commonly accepted flow timeout is 64 seconds as proposed in [5], i.e., if no packet arrives in a specific flow for 64 seconds, the flow expires. Discriminators such as host distribution and traffic volume are used to characterize the traffic. However, there are situations that flow-based discriminators are considered insufficent, e.g., when deploying TBNAN for P2P file sharing node identification (see Section **3.1**). In these cases, we move beyond the flow-level to the host-level, where the discriminators could depict an overall picture of the communication. To do so, we treat all the communication bounded to a target host as a single stream and extract discriminators upon it. In TBNAN, the primary incoorporated tool that supports collection of network traces is the portable libpcap C/C++ library[6]. It also supports analysis on flows that are generated from NetFlow or sFlow that are industry standard technologies for monitoring high-speed switched networks.

2.2 Computation server

The computation server creates prediction models by deploying data mining engines to received data. In TBNAN, we are currently making use of the following three types of analysis engines.

2.2.1 Statistical classification

Statisitical classification is a process to categorize network streams into known classes based on their statistical characteristics. Depending on the access point to the network and the purpose of the learning, TBNAN deploys different classification engines on different discriminators. See Section **3.1** for a case study on how to identify a user who is using P2P file sharing clients from regular users. See Section **3.2** for a case study on how to classify the network flows into normal network traffics and cyber attacks.

2.2.2 Anomaly detection

Anomaly detection refers to detecting patterns in the given data that do not conform to an established normal behavior[8] [9]. The abnormal patterns thus detected are called anomalies and often translate to critical and actionable information in the application domains. The anomaly detection engine of TBNAN builds a model of normal network streams and detects deviations from the normal model in observed streams and predicts them as anomaly states. See Section **3.3** for an example of anomaly detection to identify network scans.

2.2.3 Clustering

Clustering or cluster analysis is the assignment of a set of observations into subsets (also called clusters) so that observations in the same cluster are similar measured by some similarity metric. Clustering is a good way for quick review of data, by reducing the number of items in the data and providing a simple profile of individuals. See Section **3.4** for an

example of clustering on network attack type analysis.

2.3 Database and web server

The database and web server provides user with the indexing and visualization services of the imported data as well as the analysis results, e.g., graphics and summary reports, acquired from the computation server. One of the key points of this server is how to organize the data in the storage so that performance of answering a user query could be optimized. Database management techniques are widely studied in various fields of computer science and engineering and are beyond the scope of this paper.

3 Analysis engines and case studies

This chapter presents the case studies and performance evaluation of different analysis engines for various cyber security purposes.

3.1 P2P file-sharing node classification

Recent statistical studies on telecommunication networks outline that peer-to-peer (P2P) file sharing is keeping increasing and it now contributes about 50–80% of the overall Internet traffic[7]. Moreover, more and more network applications such as streaming media, Internet telephony, and instant messaging are taking a form of P2P telecommunication. The bandwidth intensive nature of P2P applications suggests that P2P traffics can have significant impact on the underlying network. Therefore, analyzing and characterizing this kind of traffic is an essential step to develop workload models towards efficient amelioration in network traffic engineering and capacity planning. This section presents the application of classification techniques for purpose of P2P file sharing node identification.

As aforementioned, our analysis makes use of discriminators extracted only from packet headers. By only using header information, we mitigate the collection and computation costs of alternative signature based approaches, and avoid privacy-related issues that could be caused by inspection into the traffic payload.

Traffic characteristics on metrics such as the traffic volume, packet size and number of preserved connections are often good indicators of P2P applications. Previously, these features are often derived on network flows. However, recent P2P applications tend to hide their appearances by disguising as ordinary network transmission such as web browsing or FTP file transferring. Therefore, flow level statistics could not offer sufficient information for discriminating a P2P flow from flows associated to ordinary client-server type applications. In this case study we are more interested in the status of a host, i.e., whether it is doing P2P file sharing, rather than the characteristics of its specific communication session. For better capturing the decentralized nature of a P2P network, we go beyond the flow-level to the host-level, where an overall picture of the communications is readily available. To do so, we treat all the communications bounded to a target host as a single stream and discriminators are defined upon streams collected from different hosts.

To collect the traces, TBNAN is applied to the virtualization based P2P traffic creation system introduced in [10], which supports inexpensive data labeling for the captured P2P traces. The system performs classification between the background traffic and two most popular P2P protocols, i.e., BitTorrent, which is the world's most popular P2P file sharing protocol, and PPLive, which is a typical protocol of the new generation P2P applications known as P2PTV. The training and testing are performed on the trace data that are collected in the same network environment.

In the first experiment, we explore the influence of the time-window size, w, on the generalization performance of the system. To justify whether a client is a P2P node, traffic related to this client need to be monitored for at least a time period of w-seconds so that discriminators could be extracted on the captured trace during this period. Training and prediction are made based on these discriminators.

In this sense, the size of the time window is closely related to the response performance of the system. In the experiment, w is selected from {1, 2, 4, 8, 16, 32, 64} seconds. As w changes its value, the variation of classification accuracy on the test set is recorded and shown in Fig. 2(a). As shown in the figure, along with the increment of w, discriminant information in the discriminitors gradually increases. When the sampling rate is 1, i.e., all captured data are used for feature extraction, the classification rate is increased from 95.56% (w = 1s) to 99.60% (w = 64s). Lowering the sampling rate, which is noted as an r parameter hereinafter, leads to a degeneration of the accuracy to some extent. Still, in all cases, the accuracy increases as the window size grows. When w = 64s, the accuracy approaches 99.53% for a sampling rate of 1 and 97.44% for a sampling rate at 1/8, respectively.

When monitoring high speed switched networks, the sampling rate r on the network traces is another important parameter that determines the scalability of the monitoring-system. Generally, we want to reduce the sampling rate for purpose of cost effective traffic data collection, storage, and analysis. The second group of experiments are designed to verify the influence of sampling rate on the prediction accuracy. To do so, the traces are first captured using full sampling, i.e., r = 1, followed by a subsampling procedure using different r parameters selected from {1/64, 1/32, 1/16, 1/8, 1/4, 1/2, 1}. Then discriminators are extracted from the subsampled traces and forwarded to the classifier for training. Figure 2(b) shows the curves of prediction accuracy against sampling rate on the test set. The figure looks similar to that of Fig. 2(a). At this time, the increment in sampling rate contributes to the increment in the prediction accuracy. For w = 1s, the accuracy starts from 85.43% at r = 1/64, slightly drops to a minimum of 83.71% at r = 1/8, and then increases gradually to 95.56% at r = 1. For w = 8s, the accuracy constantly increases from 88.34% at r = 1/64 to 99.37% at r = 1. For w = 64s, except the beginning point at r = 1, all other r values all give accuracies above 99.00%.

To summarize, the above experiments on windows size and sampling rate show that, more computation resources, i.e., longer observation time and/or higher sampling rate will let us know better about the behavior of the client, and thus results in higher prediction accuracy – as typically experienced in all computer science related fields. Yet a very important discovery is that, the positive effect of increasing the window size of observation excesses the negative effect of decrement in sampling rate. This suggests that for better recognition accuracy and less system performance deteriation, we can keep a rather small sampling rate for less network performance influence

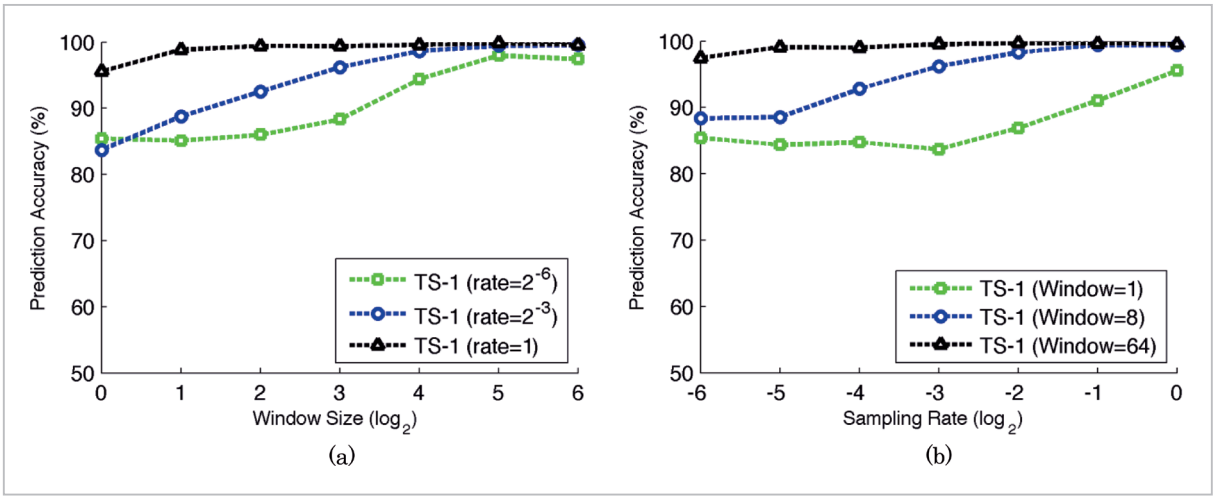


Fig.2 *Prediction accuracy v.s. window size*

but increase the window size untill satisfactory generalization performance is guaranteed. Aboveall, $w = 64$s and $r = 1/32$ seems to be a very good parameter combination that not only supports good accuracy on prediction of P2P users (99.04% on the test set) but also imposes a low cost of network resources.

3.2 Network attack classification by HMEB

In this section, by moving from the host-level analysis to the flow-level analysis, we present the application of Hierarchical Minimal Enclosing Balls (HMEB)[11] for multi-label intrusion detection classification, as a case study on flow-level network traffic classification.

The study on HMEB is inspired by the fact that network intrusions are usually associated with multiple labels as in a hierarchical structure. Consequently, the classification of network intrusions could naturally be formulated as a hierarchical multi-label classification (HMC) problem[12], in which every instance may belong to more than one class. Previous researches generally focus on exclusively grouping network attacks into several major clusters, thereby they may reduce the detection efficiency because of the loss of important lower level information of intrusion attacks.

Figure 3 gives an example of HMEB data approximation over a synthetic dataset with a two-layer hierarchy, where multi-label $A_1 \supset A_2, A_3, A_4$, i.e., A_1 is the parent class of A_2, A_3 and A_4. HMEB builds Minimal Enclosing Balls (MEBs) that separates from or encompassing with each other, forming a tree-like classification structure. Given an unlabeled sample, HMEB seeks a MEB that encloses the sample, and labels the sample according to the MEB's position in the MEB hierarchy.

To apply the HMEB algorithm to the collected network traces, flow-level discriminators are extracted and labeled with correct class labels, with some, or all, of the samples belonging to multiple classes. Then the samples together with their labels are formulated as an HMC problem and solved by HMEB. To compare the performance of HMEB with other methods, we make used of the KDD'99 intrusion detection dataset, which is a widely used benchmarking repository in the field.

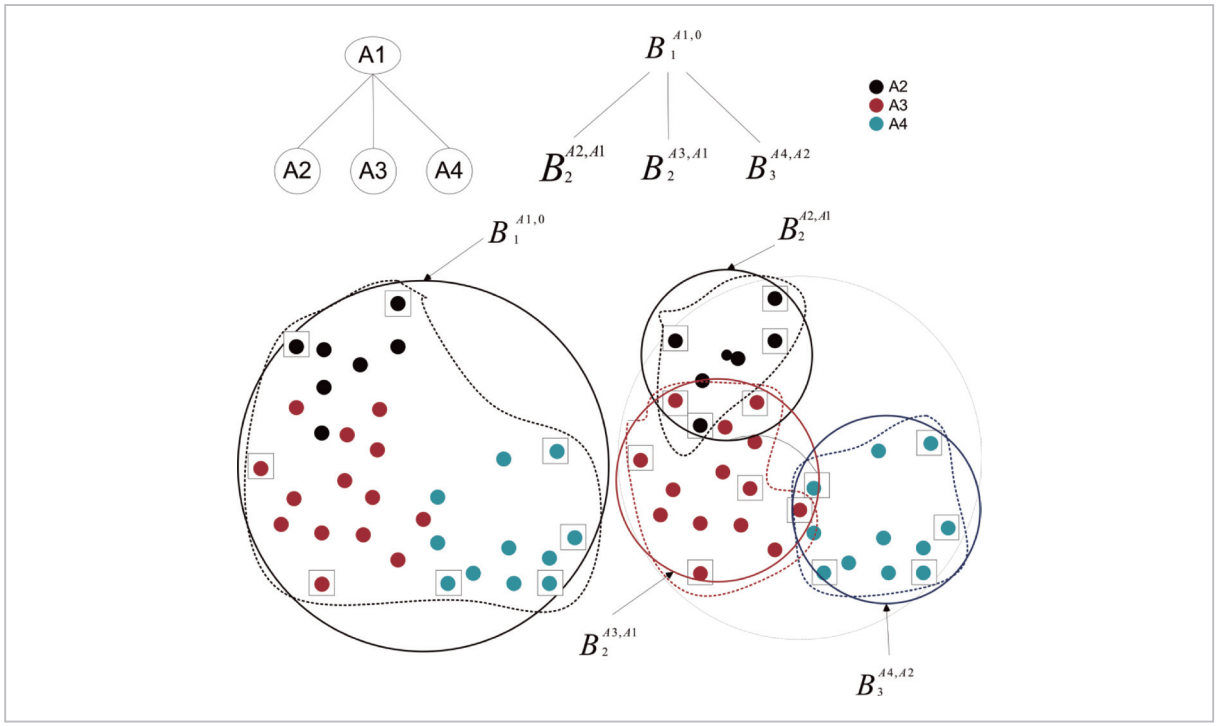


Fig.3 HMEB structure on a 2D example

As seen from Table 1, Bernhard (1999) achieved an extremely high classification accuracy of 99.5% on normal connect type, however his classifier showed poor classification performance on U2R and R2L, with none of them exceeding 15% because the class size of these two classes are significantly smaller than the other classes. Although the overall classification accuracy of HMEB is slightly lower than that of the Bernhard's method for normal connection type and Denial of Service (DoS) attacks, HMEB beats Bernhard's method on classification of two most important classes. Particularly, HMEB increases the classification accuracy of U2R and R2L by 70% and 35%, respectively. This result shows the advantage of HMEB: by enclosing all the samples of a corresponding class, HMEB enables a more accurate approximation of class boundary, even for skewed classes.

As a summary, experimental results show that HMEB has significant improvement over previously published benchmark works. In particular, HMEB improves U2R accuracy from 13.2% to 82.7% and that of R2L from 8.4% to 45.9%, as compared to the winner of KDD'99. It is also noteworthy that results of HMEB are obtained with less computation costs, as the computational time stays steady while the size of training data may exponentially manifold.

3.3 Anomaly detection engine

The second data mining tool embedded in TBNAN is the anomaly detection engine. TBNAN applies one class SVM algorithm (OCSVM)[13] for anomalous network status detection. The OCSVM algorithm first maps input data into a high dimensional feature space (via the so-called kernel trick) and then finds the maximal margin hyperplane that best separates the training data from the origin by solving a quadratic optimization problem. When mapped back from the kernel-induced feature space, the hyperplane will corresponds to the cluster boundaries in the input space.

Figure 4 depicts an example that anomaly detection is applied to the observed streams (at host-level) to detect abnormal status corresponding to network port scan, as a case study for applying anomaly detection with TBNAN. This method builds a model of normal network streams, detects deviations from the normal model in observed streams, and reports them as anomaly states. In this case study, the OCSVM is based on the following two discriminators: the number of source ports that have appeared in a unit time frame (x-axis) and the type of TCP-Flags that have appeared

Table 1 *Classification accuracy comparison with Bernhard (1999)*

Actual VS Predicted	Actual Normal	Actual DOS	Actual U2R	Actual R2L	Actual Probe
Bern. predicted Normal	60262	5299	168	14527	511
HMEB predicted Normal	920431	36818	9845	2603	2084
Bern. predicted DOS	78	223226	0	0	184
HMEB predicted DOS	153064	3619301	10341	14732	85932
Bern. predicted U2R	4	0	30	8	0
HMEB predicted U2R	6	3	43	0	0
Bern. predicted R2L	6	0	10	1360	0
HMEB predicted R2L	321	193	44	517	0
Bern. predicted Probe	243	1328	20	294	3471
HMEB predicted Probe	2843	2104	412	506	35237
Bern. total accuracy	99.5%	97.1%	13.2%	8.4%	83.3%
HMEB total accuracy	94.6%	93.2%	82.7%	45.9%	85.7%

in the same time window (y-axis). Anomaly spots can be detected during the port scan in reference to the decision function indicated by the threshold plane (with normal traffic under the plane and port scan above the plane) in the figure.

3.4 Clustering

The next analytical tool in TBNAN is clutering. Clustering helps to give a concise presentation of the traffic patterns for better understanding the large-scale repositories created by continuous monitoring over an access point with massive throughput. Literature shows that despite of its advantages, the classic clustering technique, *K*-means, suffers from certain shortcomings that prevent its application for practical network trace analysis. Especially, the value of *K*, i.e., the number of clusters, in *K*-means is determined by heuristics, which generally cast a shadow on the detection accuracy. In the following, a heuristic clustering algorithm called *G*-means[14] is presented for intrusion detection. *G*-means makes use of a density-based clustering method as a preprocessing step to select appropriate *K* parameter, and thus addresses the aforementioned problem of *K*-means.

G-means can be divided into two phases: training phase to build clusters and detecting phase to detect intrusions. At the first step of the training phase, a density-based clustering algorithm is performed on the input data to obtain the cluster ordering and information such as cluster membership of each sample. In our study, the OPTICS algorithm[15] is adopted because of its efficiency. Then, the number of clusters and the member of kernels from the ordering are extracted from the clustering results, where a kernel is the data points at the core of a cluster. Finally, *K*-means clustering is performed with subsidiary information from the previous step. In the detecting phase, detection of intrusions is performed by assigning an unknown data into the cluster whose centroid is the closest to the query instance, and then a data sample which is too distant from the centroid is reported as anomalous.

In the following, we evaluate the perfor-

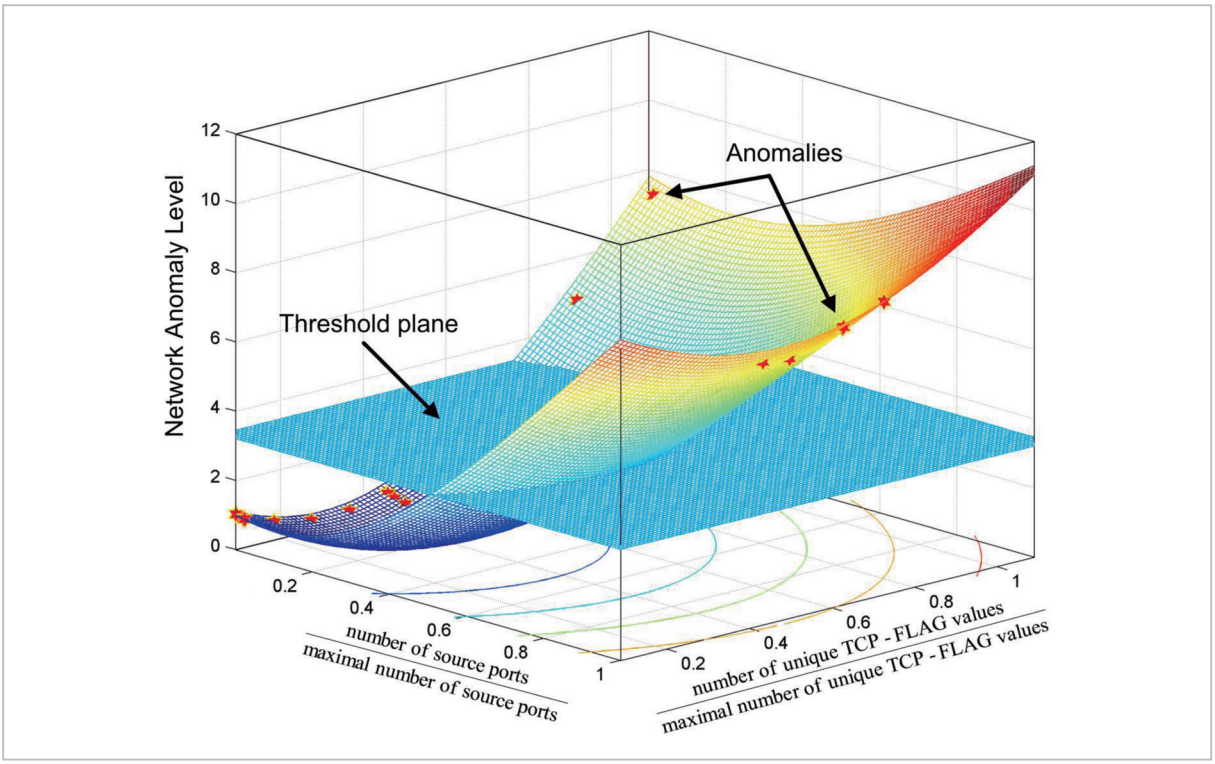


Fig.4 *Network anomalous status detection by OCSVM*

Table 2 *Comparison in Detection Rate (DR) and False Positive Rate (FPR)*

	G-means			*K*-means			OPTICS
# clusters	621	5	23	300	600	700	621
DR	99.12%	96.2579%	97.5292%	98.9820%	99.1953%	99.1166%	94.6456%
FPR	1.4107%	0.3264%	3.0951%	0.9001%	1.5054%	1.9213%	24.5815%

mance of *G*-means, *K*-means, and OPTICS on a subset of 100,000 samples randomly sampled from the KDD'99 dataset. *K*-means is run on different *K* values. The comparison results are shown in Table 2. In the table, *G*-means obtained satisfactory results for the three parameter-settings, while *K*-means requires multiple runs with different *K* values to figure out the best *K*. Both *G*-means and *K*-means outperform OPTICS on detection rate and false positive rate. With a closer inspection to the result from OPTICS, we find that its defficiency in detection rate is mainly caused by suboptimal partition of the dataset that in turn leads to ignorance of some training samples as noises.

To summarize, the experiment results show that *G*-means is effective for intrusion detection, producing a high detection rate and a low false positive rate. It is also able to automatically reveal the number of clusters in the dataset and give reasonable initialization to the cluster centroids, which makes *G*-means converge faster at betteer optimum conditions than *K*-means.

4 Conclusion

TBNAN comprises a suite of data mining algorithms, which can be used as a tool by network analysts to defend the network against emerging cyber threats and possible incidents. The various components of TBNAN such as the host-level and flow-level classification engines, the anomaly detection based scan detector, and the *G*-means clustering engine could help to detect different types of attacks and intrusions on a computer network or to facilitate analytical purpose such as the behavior analysis of network users and network applications. In addition to the analysis engines described in this paper, we are now positively incorporating other intelligent analytical tools for better understanding the nature of the monitored network environment and prevention of misuses and cyber threats.

References

1 Barbar, D. and Jajodia, S. (eds.), "Applications of Data Mining in Computer Security," Kluwer, Dordrecht, 2002.

2 Chan, P. K. and Lippmann, R. P., "Machine learning for computer security," Journal of Machine Learning Research, 7, pp. 2669–2672, 2006.

3 Maloof, M. (ed.): "Machine Learning and Data Mining for Computer Security," Springer, Heidelberg, 2006.

4 Inoue, D., Yoshioka, K., Eto, M., Yamagata, M., Nishino, E., Takeuchi, J., Ohkouchi, K., and Nakao, K., "An Incident Analysis System NICTER and Its Analysis Engines Based on Data Mining Techniques," LNCS, 2009, Volume 5506, pp. 579–586, 2009.

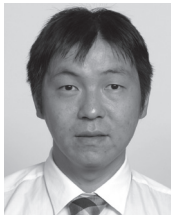
5 Claffy, K., Braun, H. W., and Polyzos, G., "A Parametrizable Methodology for Internet Traffic Flow Profiling," IEEE JSAC, 13(8): 1481–1494, 1995.

6 http://www.tcpdump.org/

7 http://www.ipoque.com/news & events/internet studies/internet

8 Kriegel, H. P., Kroger, P., and Zimek, A., "Outlier Detection Techniques (Tutorial)," PAKDD, Bangkok, Thailand, 2009.
9 Chandola, V., Banerjee, A., and Kumar, V., "Anomaly Detection: A Survey," ACM Computing Surveys, Vol. 41(3), Article 15, 2009.
10 Ban, T., Ando, R., and Kadobayashi, Y., "Monitoring and Analysis of Network Traffic in P2P Environment," Journal of the National Institute of Information and Communications Technology, Vol. 55, Numbers 2/3, 2008.
11 Chen, Y., Pang, S., Kasabov, N., Ban, T., and Kadobayashi, Y., "Hierarchical Core Vector Machines for Network Intrusion Detection," ICONIP 2009, Part II, LNCS 5864, pp. 520–529, 2009.
12 Boutell, M. R., "Learning multi-label scene classification," Pattern Recognition, Vol. 37, No. 9, pp. 1757–1771, 2004.
13 Scholkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., and Williamson, R., "Estimating the support of a high dimensional distribution," Neural Computation, 13(7): 1443–1472, 2001.
14 Zhao, Z., Guo, S., Xu, Q., and Ban, T., "G-Means: A Clustering Algorithm for Intrusion Detection," ICONIP 2008, Part I, LNCS 5506, pp. 563–570, 2009.
15 Mihael, A., Markus, M. B., Hans-Peter, K., and Jorg, S., "OPTICS: Ordering Points to Identify the Clustering Structure," ACM SIGMOD 1999 International Conference on Management of Data, pp. 49–60, ACM Press, Philadelphia, 1999.

(Accepted June 15, 2011)



BAN Tao, *Ph.D.*

Expert Researcher, Cybersecurity Laboratory, Network Security Research Institute

Cybersecurity, Data Mining

KADOBAYASHI Youki, *Ph.D.*

Guest Expert Researcher, Network Security Research Institute/Associate Professor, The Graduate School of Information Science, Nara Institute of Science and Technology

Cybersecurity, Internet Engineering