A FLOW BASED APPROACH FOR SSH TRAFFIC DETECTION

Riyad Alshammari , Member, IEEE , and A. Nur Zincir-Heywood , Member, IEEE

Abstract— **The basic objective of this work is to assess the utility of two supervised learning algorithms AdaBoost and RIPPER for classifying SSH traffic from log files without using features such as payload, IP addresses and source/destination ports. Pre-processing is applied to the traffic data to express as traffic flows. Results of 10-fold cross validation for each learning algorithm indicate that a detection rate of 99% and a false positive rate of 0.7% can be achieved using RIPPER. Moreover, promising preliminary results were obtained when RIPPER was employed to identify which service was running over SSH. Thus, it is possible to detect SSH traffic with high accuracy without using features such as payload, IP addresses and source/destination ports, where this represents a particularly useful characteristic when requiring generic, scalable solutions.**

I. INTRODUCTION

One of the important tasks of network management is accurate identification and classification of network traffic. For example, a network administrator may want to identify and throttle/block traffic from different applications to manage bandwidth budget and to ensure quality of service objectives are met for business critical applications. Similar to network management tasks, many network engineering problems such as workload characterization, and modeling, capacity planning, traffic shaping/policing and route provisioning also rely on accurate classification of network traffic. On the other hand, an accurate method for reliably identifying the applications associated with network traffic is still to be developed.

One approach to classifying applications is to inspect the payload of every packet. This technique can be very accurate when the payload is not encrypted. However, this is not realistic in practice. Firstly, there are privacy concerns with examining user data. Secondly, there is a high computational and storage overhead when attempting to study every packet that traverses a link; especially given the wide spread dependency on high-speed links. Thirdly, there are applications such as SSH, which has encrypted payload, implying that the payload is opaque. Thus, applications have traditionally been classified using well-known TCP/UDP port numbers. However this approach has become increasingly inaccurate. Mostly because applications use non-standard ports to avoid detection or to circumvent operating system restrictions. Furthermore, ports can be dynamically allocated as needed, i.e. the same port number can be used to transmit multiple applications. Thus, other techniques are needed to increase the accuracy of network traffic classification.

Research in this area have employed various classification techniques, section II. In general all these efforts show that even though it is easy to apply such techniques to well known application traffic such as WWW, more work is needed to identify applications that can run different services over the same port such as SSH.

SSH is typically used to login to a remote computer but it also supports tunneling, file transfers and forwarding arbitrary TCP ports over a secure channel between a local and a remote computer. What makes the detection of this application interesting is that its traffic is encrypted. Thus, any payload analysis based classification technique becomes irrelevant since the payload is encrypted. Moreover, multiple services/applications can be run on an SSH session. This makes identifying SSH very important in terms of network engineering and management tasks. In general, SSH accounts for much less traffic than other applications (around 1 to 2%) in a given traffic log file (see NLANR [8] and MAWI [9] repositories). This in return results in very unbalanced data sets for SSH detection.

Thus, the objective of this work is to examine the utility of two machine learning algorithms, namely AdaBoost, and RIPPER to solve this classification task without using IP addresses, port numbers and payload information. To do so, features will be derived from flow-based information for the automatic recognition of SSH traffic and the applications running over it.

The remainder of the paper is organized as follows. Section II establishes the position of this work, relative to earlier research. The machine learning based approaches are presented in section III. The methodology for automating ssh classification is discussed in section IV. Experimental results are presented in section V, and conclusions are drawn in Section VI.

II. BACKGROUND

Most of the research existing in the literature detailing traffic detection approaches focuses on automatic application recognition. To the best of our knowledge, all but two [13] , [14] of the previous work require either payload inspection or employ information about the flows including the port numbers to automate the application recognition.

Zhang and Paxson present one of the earliest studies of techniques based on matching patterns in the packet payloads [10]. Dreger et al. [11] and Moore et al. [4] applied more sophisticated analyses, which still require payload inspection. Early et al. employed a decision tree classifier on *n*-grams

Manuscript received March 16, 2007. This work is supported in part by MITACS, CFI , and NSERC Discovery grants.

R. Alshammari is with Dalhousie University, Faculty of Computer Science, Halifax, NS B3H 1W5, Canada. (e-mail:riyad@cs.dal.ca)

A. N. Zincir-Heywood is with Dalhousie University, Faculty of Computer Science, Halifax, NS B3H 1W5, Canada (phone: 902-4943157; fax: 902-4921517;e-mail: zincir@cs.dal.ca)

of packets for distinguishing flows [12]. Moore et al. used Bayesian analysis to classify flows into broad categories such as bulk transfer, P2P or interactive [3] , [4]. Haffner et. al. employed AdaBoost, Hidden Markov, Nave Bayesian and Maximum Entropy models to classify network traffic into different applications [2]. They found that AdaBoost performs the best. Their results show that they can classify ssh with 86% detection rate and 0% false positive rate but they employ the first 64 bytes of the application (payload). Bernaille et al. employed k-means clustering to the first five packets in each connection to identify application protocols [6]. On the other hand, Karagiannis et al. proposed an approach that does not use port numbers or payload information [5]. However, their approach relies on information about the behavior of the hosts on the network. Specifically, they make use of social and functional roles of hosts, that is, their interactions with other hosts and whether they act as a provider or consumer of a service, respectively. Thus, their approach may be capable of detecting the type of an application, but might not be able to identify distinct applications and it does not classify individual flows or connections [13]. More recently, Wright et al. investigate the extent to which common application protocols can be identified using only packet size, timing and direction information of connection information [1] , [13]. They employed a k Nearest Neighbor (kNN) and Hidden Markov Model classifiers to compare the performance. Even though their approach can classify distinct encrypted applications, their performance on SSH classification is (76% detection rate and 8% false positive rate) is not as good as well known application traffic such as WWW and instant messaging. Another recent work by Williams et al. [14] compared five different classifiers namely, Bayesian Network, C4.5, Nave Bayes (with discretisation and kernel density estimation) and Nave Bayes Tree, on the task of traffic flow classification. They found that C4.5 performed better than the others. However, rather than giving classification results per application, they give overall accuracy results per machine learning algorithm. We believe that this is misleading especially on unbalanced data sets where say only 5% of the data set is in-class and 95% out-class. Then by just labeling everything as out-class, a classifier can achieve 95% accuracy, but actually misclassifies all of the in-class exemplars.

Finally, to the best of our knowledge, none of the above work attempted to classify/identify applications/services running over SSH. Thus, in order to classify SSH traffic, other approaches must be taken. In this work, we will apply two machine learning approaches and see how far we can take it for this purpose.

III. MACHINE LEARNING BASED APPROACHES

In this work, in order to classify SSH traffic, two different machine learning approaches will be analyzed. These are RIPPER, and AdaBoost.

A. RIPPER

RIPPER, Repeated Incremental Pruning to Produce Error Reduction, is a rule based machine learning algorithm [15]. Rules are added to explain positive examples such that if an instance is not covered by any rule then it is classified as negative. In RIPPER, conditions are added to the rule to maximize an information gain measure [16]. Change in gain is defined as shown in equation 1.

$$Gain(R^{'}, R) = s.(log_2 \frac{N_+^{'}}{N^{'}} - log_2 \frac{N_+}{N}) \quad (1)$$

where N is the number of instances that are covered by R and $N+$ is the number for true positives in them. $N^{'}$ and $N^{'}{}_+$ are similarly defined for $R^{'}$. s is the true positives in R and $R^{'}$ after adding the condition. In short, the change in gain measures the reduction in bits to encode a positive instance. Conditions are added to a rule until it covers no negative examples. Once a rule is grown, it is pruned back by deleting conditions in reverse order to find the rule that maximizes the rule value metric, equation 2.

$$rvm(R) = \frac{p-n}{p+n} \quad (2)$$

where p and n are the number of true and false positives, respectively. To measure the quality of a rule, minimum description length is used [16]. RIPPER stops adding rules when the description length of the rule base is 64 (or more) bits larger than the best description length. A more detailed explanation of the algorithm can be found in [16].

B. AdaBoost

AdaBoost, Adaptive Boosting, was developed by Freund and Schapire [17]. It is a meta-learning algorithm, which means that a strong classifier is built from a linear combination of weak (simple) classifiers. It incrementally constructs a complex classifier by overlapping the performance of possibly hundreds of simple classifiers using a voting scheme. These simple classifiers are called decision stumps. They examine the feature set and return a decision tree with two leaves. The leaves of the tree are used for binary classification and the root node evaluates the value of only one feature. Thus, each decision stump will return either +1 if the object is in class, or 1 if it is out class. Fig. 1 summarizes the algorithm. AdaBoost is simple to implement and known to work well on very large sets of features by selecting the features required for good classification. It has good generalization properties. However, it might be sensitive to stopping criterion or result in a complex architecture that is opaque. A more detailed explanation of the algorithm can be found in [16].

IV. METHODOLOGY

We divided the problem of classifying SSH traffic into two phases. In the first phase, we will concentrate on the automatic recognition of SSH traffic from a given traffic file. In the second phase, we will concentrate on developing the automatic recognition of different services/applications

TRAINING:
For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = \frac{1}{N}$
For all base-learners $j=1,...,L$
Begin
Randomly draw X_j from X with probabilities p_j^t
Train d_j using X_j
For each (x^t, r^t), decrease probabilities if correct:
Begin
If $y_j^t = r^t p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$
End
Normalize Probabilities: $Z_j \leftarrow \sum_t p_{j+1}^t$; $p_{j+1}^t \leftarrow \frac{p_{j+1}^p}{Z_j}$
End

TESTING:
Given x, calculate $d_j(x)$, $j=1,...,L$
Calculate class outputs, $i=1,...,K$: $y_i = \sum_{j=1}^L (\log \frac{1}{\beta_j}) d_{ji}(x)$

Fig. 1. AdaBoost Algorithm [16]

running over the SSH, such as ssh, scp, sftp, tunnel and xForward. In both cases, we are going to determine the application without using payload, IP addresses or port numbers based on flow based information only.

As discussed earlier, we are going to use two machine learning algorithms (RIPPER, AdaBoost) in order to identify the best fitting one to the problem. The reason we choose these algorithms are two fold. Haffner et. al. employed AdaBoost, Hidden Markov, Nave Bayesian and Maximum Entropy models to classify network traffic into different applications [2]. In their work, they conclude that AdaBoost gives the best performance results. Thus, we select AdaBoost in order to provide the performance baseline in the problem domain of this work. On the other hand, RIPPER is known to be superior in information retrieval tasks [18]. In addition, we have preliminary results for the analysis of RIPPER algorithm applied to the second phase of the project.

A. Data Collection

In our experiments the performance of the different machine learning algorithms is established on two different data sources: public and private. We are going to use the public data sets for the first phase and private data sets for the second phase.

Public data sets, are traffic traces captured on real networks, naturally have no reference to payload data or network configuration because of privacy issues. Since our classifiers are not based on payload analysis, we use several public data sets from NLANR [8] and MAWI [9] repositories. NLANR data sets consist of Time Sequenced Header (TSH) files while MAWI data sets are in tcpdump file format. A TSH file represents each packet with 44 bytes that include a timestamp, interface number, IPv4 header without options, and the first 16 bytes of the transport header.

Private data sets consist of packets captured internally at our research testbed. Our data collection approach is to simulate possible network scenarios using one or more computers to capture the resulting traffic. We simulate an SSH connection by connecting a client computer to four SSH servers outside our testbed via the Internet. We run the following six SSH services: (i) Shell login; (ii) X11; (iii) Local tunneling; (iv) Remote tunneling; (v) Scp; (vi) Sftp. We also run and captured some other services (background traffics) such as DNS, HTTP, FTP, P2P (limewire), and telnet.

TABLE I
FLOW BASED FEATURES EMPLOYED

Min forward packet length	Std. deviation of forward inter-arrival times
Mean forward packet length	Min backward inter-arrival time
Max forward packet length	Mean backward inter-arrival time
Std deviation of forward packet length	Max backward inter-arrival time
Min backward packet length	Std. deviation of backward inter-arrival times
Mean backward packet length	Protocol
Max backward packet length	Duration of the flow
Std. deviation of backward packet length	# Packets in forward direction
Min forward inter-arrival time	# Bytes in forward direction
Mean forward inter-arrival time	# Packets in backward direction
Max forward inter-arrival time	# Bytes in backward direction

B. Feature Selection

Based on the above model, we represented the traffic using flow-based features. These representations are used as the input vector to the machine learning algorithms. In this case, each network flow is described by a set of statistical features and associated feature values. Here, a feature is a descriptive statistic that can be calculated from one or more packets.

We used NetMate [19] tool set to process data sets, generate flows and compute feature values. Flows are bidirectional and the first packet seen by the tool determines the forward direction. Moreover, flows are of limited duration. UDP flows are terminated by a flow timeout. TCP flows are terminated upon proper connection teardown or by a flow timeout, whichever occurs first. There is not a general and accepted value for the flow time out [20] , [21] , [22],[23],[24],[25],[26]. Therefore, we decide to use a 600 second flow timeout value; where this corresponds to the IETF Realtime Traffic Flow Measurement working groups architecture [27]. This is also the value used in [14]. Finally, we consider only UDP and TCP flows that have no less than one packet in each direction and transport no less than one bytes of payload. We extract the same set of features (without employing any feature reduction techniques) used in [14] to provide a comparison environment for the reader, table I. Again, features such as IP addresses and source/destination port numbers are excluded from the feature set to ensure that the results are not dependent on such biased features[1] .

C. Labeling Data Sets

With public data sets, there is no accurate method of determining the type of application to which different packets

[1]In other words, the classification will not be based on trivial solutions in which the ports of the machines in the data set or their IP addresses are correlated with class labels.

belong, since we do not have access to packet payloads. Thus, we simply use the same method (a port based classifier) as used in [2] , [4], [7], [13], [14]. Naturally, as destination and source port addresses are not part of the feature vector, successful machine learning solutions must find a more generic alternative classification rule. On the other hand, classifying private data sets could be done accurately because we know exactly which applications were running in every simulation.

V. RESULTS AND DATA SETS

In traffic classification, two metrics are typically used in order to quantify the performance of the classifier: Detection Rate (DR) and False Positive Rate (FP). A high DR rate and a low FP rate would be the desired outcomes. They are calculated as follows:

$$DR = 1 - \frac{(\#FNClassifications}{TotalNumberSSHClassifications)}$$

$$FP = \frac{\#FPClassifications}{TotalNumberNon - SSHClassifications}$$

where FN, False Negative, means SSH traffic classified as non-SSH traffic and FP means non-SSH traffic classified as SSH traffic. Once the aforementioned feature vectors are prepared for the data sets, then RIPPER, and AdaBoost classifiers are trained on the data using a 10-fold cross validation. To this end, we have used Weka [28], which is an open source tool for data mining tasks. Weka provides an easy to use interface for the two different machine learning algorithms we employ. We employed Weka with its default parameters to run RIPPER, and Adaboost on our data sets.

A. Data Sets

We concentrated on selecting traffic traces captured at different locations and in different years on the Internet. Thus, we use data obtained from several traffic traces (data sets) in order to increase the quality of the training performance. Since we want to have SSH flows that fulfill the flow criteria, i.e. TCP or UDP flows that have at least one packet in each direction and transport one byte of payload, we use public traffic traces from the NLANR [8] and the MAWI Working Group Traffic Archive [9], table II.

In these data sets, we observe that the more SSH traffic in a given trace, the lower the contribution from FTP and TELNET traffic is in such a trace. Therefore, we decided to sample two data sets (SMALL and LARGE) from these traces that consist of a number of prominent applications. The first data set is generated to investigate the performance of the classifiers in differentiating the SSH traffic from other traffic. The second data set is generated to examine the behavior of the classifiers on a large sample when the flow features are implemented.

The first data set (SMALL) is generated by sampling 100 randomly selected (uniform probability) flows of six classes (FTP, SSH, TELNET, MAIL, DNS, HTTP) from 8 NLANR and 5 MAWI traces (200604061400, 200606061400, 200604121400, 200603161400 and 200605231400) employed. The flows are then aggregated into a single data set containing 600 flows per trace. Thus, this creates a total of 7,800 instances in the data set. On the other hand, the second data set (LARGE) is generated by sampling 2000 randomly selected (uniform probability) flows of four classes (SSH, MAIL, DNS, HTTP) from two NLANR traces (20050313 and 20050314) and six MWAI traces (200604121400, 200605141400, 200606111400, 200603161400, 200601291400, 200603291400). The flows are then aggregated into a single data set containing 8,000 flows per trace. The choice of eight traces creates a total of 64,000 instances in the data set. Once the above data sets are generated, we ran RIPPER, and AdaBoost on these new data sets with NetMate Flow based features.

TABLE II
TRAFFIC TRACES USED FROM THE PUBLIC DATA REPOSITORIES

NLANR (AMPATH-I)	MAWI
20050312	200604061400
20050313	200606061400
20050314	200604121400
20050316	200603161400
20050317	200605231400
20050318	200605141400
20050320	200606111400
20050321	200601291400
	200603291400

TABLE III
PERFORMANCE ON THE SMALL DATA SET

	RIPPER		AdaBoost	
Applications	**FP**	**DR**	**FP**	**DT**
TELNET	0.083	0.962	1	0
FTP	0.032	0.968	1	0
HTTP	0.056	0.927	1	0
SSH	0.054	0.936	1	0
DNS	0.012	0.971	0.435	0.999
MAIL	0.051	0.946	0.765	0.995

B. Results for Phase-1: Classifying SSH vs Non-SSH

Table III shows the performance of the algorithms on the SMALL data set. In this case, RIPPER algorithm can correctly classify 95% (overall accuracy) of the instances. Moreover, RIPPER can identify SSH traffic with a high DR and a low FP rate. Table IV shows the confusion matrix. It is now apparent that RIPPER can classify 1217 of the SHH flows correctly, whereas 22 SSH flows are misclassified as HTTP, and 15 are misclassified as Telnet. Table V shows the performance of the two algorithms on the LARGE data set. This time, RIPPER correctly classifies 99% (overall accuracy) of the instances in the data set. On the other hand, AdaBoost algorithm can only classify 50% of the instances correctly. However it cannot classify SSH flows at all. Table VI shows the confusion matrix for both algorithms on the LARGE data set, where out of 16000 SSH instances 15865 are correctly classified and only 38 of them are misclassified as HTTP and 61 of them are misclassified as Mail flows by RIPPER.

TABLE IV
CONFUSION MATRIX ON THE SMALL DATA SET

	AdaBoost					
*	C1	C2	C3	C4	C5	C6
C1	0	0	0	0	801	499
C2	0	0	0	0	28	1272
C3	0	0	0	0	62	1238
C4	0	0	0	0	104	1196
C5	0	0	0	0	1299	1
C6	0	0	0	0	6	1294
	RIPPER					
*	C1	C2	C3	C4	C5	C6
C1	1250	9	13	15	4	9
C2	6	1259	12	8	1	14
C3	34	15	1205	22	6	18
C4	36	5	20	1217	3	19
C5	18	1	10	3	1262	6
C6	19	12	16	22	1	1230

These are the applications, where C1: telnet, C2: ftp, C3: http, C4: ssh, C5: dns and C6: mail.

TABLE V
PERFORMANCE ON THE LARGE DATA SET

	RIPPER		AdaBoost	
Applications	FP	DR	FP	DT
HTTP	0.011	0.995	1	0
SSH	0.007	0.992	1	0
DNS	0.001	0.997	0.032	0.999
MAIL	0.007	0.991	0.663	0.999

In short, results show that RIPPER algorithm outperforms AdaBoost in terms of correctly classifying SSH traffic on both SMALL and LARGE flow-based data sets. The best performance is observed where RIPPER achieves 99% detection rate and 0.7% false positive rate for identifying SSH flows on the LARGE data set, which also demonstrates its capability in terms of being a good classifier. This appears to indicate that the combination of a greedy search heuristic and no provision for an unknown class category are sufficient to provide a 'good' quality classifier.

C. Results for Phase-2: Classifying Services on SSH

As discussed in section 3, a private network trace file is captured on our testbed. To this end, a data set is generated by sampling 1000 flows of eleven classes (shell login, X11, local tunneling, remote tunneling, scp, sftp, dns, ftp, http, p2p-limewire and telnet) from the private trace file. The flows are then combined into a single data set containing 11,000 flows. RIPPER and AdaBoost algorithms are run on this data set using NetMate flow based features as previously done in phase-1. Table VII shows the results of 10-fold cross validation for RIPPER and table VIII shows it for AdaBoost learning algorithms.

TABLE VI
CONFUSION MATRIX ON THE LARGE DATA SET

	AdaBoost			
Applications	HTTP	SSH	DNS	MAIL
HTTP	0	0	344	15656
SSH	0	0	172	15828
DNS	0	0	15987	13
MAIL	0	0	12	15988
	RIIPER			
Applications	HTTP	SSH	DNS	MAIL
HTTP	15912	38	7	43
SSH	76	15865	1	58
DNS	28	5	15951	16
MAIL	72	61	8	15859

TABLE VII
PERFORMANCE OF RIPPER ON THE PRIVATE DATA SET

Applications	FP	DR
TELNET	0	0.999
FTP	0.001	0.997
HTTP	0.002	0.976
DNS	0	1
P2P-LIMEWIRE	0.001	0.984
L-TUNNEL over SSH	0	0.997
R-TUNNEL over SSH	0.001	0.991
SCP over SSH	0.001	0.982
SFTP over SSH	0.001	0.989
X11 over SSH	0.001	0.984
SHELL over SSH	0.003	0.995

TABLE VIII
PERFORMANCE OF ADABOOST ON THE PRIVATE DATA SET

Applications	FP	DR
TELNET	0.605	1
FTP	0	0
HTTP	0	0
DNS	0	0
P2P-LIMEWIRE	0	0
L-TUNNEL over SSH	0.295	0.998
R-TUNNEL over SSH	0	0
SCP over SSH	0	0
SFTP over SSH	0	0
X11 over SSH	0	0
SHELL over SSH	0	0

As seen in these results, RIPPER algorithm can correctly classify 99% of the instances. Moreover, RIPPER can identify different services running over SSH with a high DR and a low FP rate (see the last six rows of table VII). On the other hand, AdaBoost cannot classify any one of the dns, telnet, ftp, p2p, http or any services running over SSH except for SSH local forwarding and TELNET flows.

D. CONCLUSION AND FUTURE WORK

In this work, we have employed RIPPER, and AdaBoost supervised learning algorithms for classifying SSH traffic from a given traffic file. To do so, we employ public data sets from NLANR and MAWI web sites based on the previous research in the general traffic classification field [14]. We tested the aforementioned learning algorithms using traffic flow based features. We have employed WEKA (with default settings) using 10-fold cross validation for each learning algorithm using each feature set.

Results so far show that RIPPER performs better than AdaBoost, on flow based features. RIPPER achieves 99% detection rate and 0.7% false positive rate at its best performance (result of 10 fold cross validation using flow based features) in detecting SSH traffic. These results are better than the ones achieved in [13] and [14], the most similar

works to ours in the literature. However, it should be noted here that the feature sets (in case of [13]) and machine learning algorithms (in both cases) employed are different as well. On the other hand, it is clear that RIPPER not only performs much better than the AdaBoost Algorithm in terms of identifying SSH among other traffic but also it seems to be a 'good' classifier in terms of its performance when it is trained on one set but tested on many others with similar statistical properties. It should be noted here that the objective of automatically identifying SSH traffic from a given traffic file is performed without using any payload, IP addresses or port numbers. Furthermore, our preliminary results on identifying services running over SSH are also promising. These results indicate that RIPPER classifier can identify interactive login sessions (SHELL), tunneling (both local and remote), SCP (secure copy), SFTP (secure file transfer) and X11 activities running over SSH.

Finally, in this work, we have shown that it is possible to identify SSH traffic from a given log file as well as classifying the applications running over SSH without using features such as payload, IP addresses and source/destination ports. Future work will follow similar lines in order to generate more realistic and larger data sets to test the robustness of the classifier for the classification of SSH and different applications running over SSH.

REFERENCES

[1] Wright C., Monrose F., Masson G. M., "HMM Profiles for Network Traffic Classification", Proceedings of the ACM DMSEC, pp 9-15, 2004.

[2] Haffner P., Sen S., Spatscheck O., Wang D., "ACAS: Automated Construction of Application Signatures", Proceedings of the ACM SIGCOMM, pp.197-202, 2005.

[3] Moore A. W., Zuev D., "Internet Traffic Classification Using Bayesian Analysis Techniques", Proceedings of the ACM SIGMETRICS, pp 50-60, 2005.

[4] Moore A., Papagiannaki K., "Toward the Accurate Identification of Network Applications", Proceedings of the Passive & Active Measurement Workshop, 2005.

[5] Karagiannis T., Papagiannaki K., Faloutsos M., "BLINC: Multilevel Traffic Classification in the Dark", Proceedings of the ACM SIGCOMM, pp 229-240, 2006.

[6] Bernaille L., Teixeira R., Akodkenou I., "Traffic Classification on the Fly", Proceedings of the ACM SIGCOMM Computer Communication Review, 2006.

[7] Erman J., Arlitt M., Mahanti A., "Traffic Classification using Clustering Algorithms", Proceedings of the ACM SIGCOMM, pp. 281-286, 2006.

[8] NLANR, , http://pma.nlanr.net/Special.

[9] MAWI, http://tracer.csl.sony.co.jp/mawi/.

[10] Zhang Y., Paxson V., Detecting back doors, Proceedings of the 9th USENIX Security Symposium, pp. 157-170, 2000.

[11] Dreger H., Feldmann A., Mai M., Paxson V., Sommer R., Dynamic application layer protocol analysis for network intrusion detection, Proceedings of the 15th USENIX Security Symposium, pp. 257-272, 2006.

[12] Early J., Brodley C., Rosenberg C., Behavioral authentication of server flows, Proceedings of the 19th Annual Computer Security Applications Conference, pp. 46-55, 2003.

[13] Wright C. V., Monrose F., Masson G. M., On Inferring Application Protocol Behaviors in Encrypted Network Traffic, Journal of Machine Learning Research, (7), pp. 2745-2769, 2006.

[14] Williams N., Zander S., Armitage G., A Prelimenary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Comparison, ACM SIGCOMM Computer Communication Review, Vol. 36, No. 5, pp. 7-15 , 2006.

[15] Cohen W. W., Fast effective rule induction, Proceedings of the 12th International Conference on Learning, pp. 115-123 , 1995 .

[16] Alpaydin E., Introduction to Machine Learning, MIT Press, ISBN: 0-262-01211-1 .

[17] Freund Y., Schapire R. E., "A Short Introduction to Boosting", Journal of Japanese Society for Artificial Intelligence, 14(5), pp. 771-780, 1999

[18] Sebastiani F., "Machine Learning in Automated Text Categorization", ACM Computing Surveys, vol. 34, no. 1, pp. 1-47, 2002.

[19] NetMate, http://www.ip-measurement.org/tools/netmate/.

[20] Caceres R., Danzig P., Jamin S., Mitzel D., Characteristics of wide-area TCP/IP conversations, of ACM SIGCOMM, pp. 101-112, 1991.

[21] Estrin D., Mitzel D., An assessment of state and lookup overhead in routers, Proceedings of IEEE pp. 2332-42, 1992.

[22] Measurement-Specs, http://www.caida.org/tools/measurement/measurementspec/.

[23] NetraMet, http://www.caida.org/tools/measurement/netramet/.

[24] Netflow, http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html.

[25] CISCO, http://www.cisco.com/en/US/products/sw/iosswrel/ps5187/products_feature_guide09186a00801b0696.html.

[26] Claffy K. C., Internet traffic characterization, PhD Thesis, University of California, San Diego, 1994.

[27] IETF, http://www3.ietf.org/proceedings/97apr/97apr-final/xrtftr70.htm.

[28] WEKA Software, http://www.cs.waikato.ac.nz/ml/weka/.