Machine Learning Based Encrypted Traffic Classification: Identifying SSH and Skype

Riyad Alshammari and A. Nur Zincir-Heywood

Abstract— **The objective of this work is to assess the robustness of machine learning based traffic classification for classifying encrypted traffic where SSH and Skype are taken as good representatives of encrypted traffic. Here what we mean by robustness is that the classifiers are trained on data from one network but tested on data from an entirely different network. To this end, five learning algorithms – AdaBoost, Support Vector Machine, Naïve Bayesian, RIPPER and C4.5 – are evaluated using flow based features, where IP addresses, source/destination ports and payload information are not employed. Results indicate the C4.5 based approach performs much better than other algorithms on the identification of both SSH and Skype traffic on totally different networks.**

I. Introduction

Many network management tasks such as managing bandwidth budget and ensuring quality of service objectives for critical applications rely on accurate classification of network traffic. Moreover, network engineering problems such as traffic shaping or workload modeling also require classification of network traffic.

Traditionally, one approach to classifying network traffic is to inspect the payload of every packet. This technique can be extremely accurate when the payload is not encrypted. However, encrypted applications such as SSH (Secure Shell) or Skype imply that the payload is opaque. Another approach to classifying applications is using well-known TCP/UDP port numbers. However, this approach becomes increasingly inaccurate when applications use non-standard ports to bypass firewalls or circumvent operating systems restrictions. Moreover, ports can be dynamically allocated as needed, i.e. the same port number can be used to transmit multiple applications, most notably port 80. Thus, other techniques are required to increase the accuracy of network traffic classification.

One possibility is to identify specific features of the network traffic and use these to guide the traffic classification. Recent research in this area focuses on the classification of efficient and effective classifiers. Different research groups have employed expert systems or various machine learning techniques such as Hidden Markov models, Naive Bayesian models, AdaBoost, or Maximum Entropy methods to this problem [1], [2], [3], [4], [5]. However, having an encrypted payload and being able to run different applications in an encrypted channel makes it a challenging problem to classify encrypted traffic from a given traffic log file. Encrypted traffic implies that performing payload analysis is not useful. Moreover port number based classification cannot be accurate given that non-standard ports can be used to avoid detection or bypass firewalls.

In this work, we have focused on the identification of two types of encrypted traffic SSH and Skype. 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. On the other hand, Skype is a proprietary P2P (Peer to Peer) VoIP network. Skype is widely known for its broad range of features, including free voice and video conferencing, and its ability to use P2P technology to overcome common firewall and NAT problems. It is a versatile method of synchronous and asynchronous communication. Indeed, covering a collection of such different encrypted behavior makes it difficult to distinguish both SSH traffic from non-SSH and Skype from non-Skype traffic. Thus, the goal of this work is to develop a model that distinguishes SSH traffic from non-SSH traffic and Skype from non-Skype traffic without using IP addresses, port numbers or payload information. We believe that this will not only enable our model to generalize from one network to another well but also potentially will enable us to employ such an approach for the classification of other encrypted applications. In order to classify/identify SSH and Skype traffic; five different machine learning algorithms will be employed. These are AdaBoost, Support Vector Machine, Naïve Bayesian, RIPPER and C4.5.

The rest of this paper is organized as follows. Related work is discussed in Section II. Section III details the data sets, features and machine learning algorithms employed. Section IV presents the experimental results. Conclusions are drawn and future work is discussed in Section V.

II. Related Work

Most of the existing research in the literature focuses on automatic application recognition in general, i.e. classification of well known applications such as HTTP, SMTP, FTP etc. However, not much attention is paid to the classification of encrypted traffic. Thus, to the best of our knowledge, most of the previous work require either payload inspection or employ information about the flows including the port numbers and IP addresses to automate the application recognition.

In the literature, Zhang and Paxson present one of the earliest studies of techniques based on matching patterns in the packet payloads [6]. Dreger et al. [7] and Moore et al. [4] applied more sophisticated analyses, which still require

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)

payload inspection. Early et al. employed a decision tree classifier on *n*-grams of packets for distinguishing flows [8]. 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, Naïve Bayesian and Maximum Entropy models to classify network traffic into different applications [2]. Their results showed AdaBoost performed the best on their data sets; with an SSH detection rate of 86% and false positive rate of 0.0%, but they employed the first 64 bytes of the payload. Since the encryption of SSH data starts after the handshake, analyzing the first 64 bytes of the payload (includes the not-encrypted part) provided them a good signature to classify SSH. However, if the encryption algorithm is changed, this system will not work correctly. Moreover, it is not generic enough to apply for other encrypted applications such as Skype or Virtual Private Networks tunnels.

Karagiannis et al. proposed an approach that does not use port numbers or payload information on traffic that is not encrypted [9]. However, their approach relies on information about the behavior of the hosts on the network. Thus, they cannot identify applications and cannot classify individual flows or connections. 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 a connection [1], [10]. They employed a k-Nearest Neighbor (kNN) and Hidden Markov Model (HMM) learning systems to compare the performance. Even though their approach can classify distinct encrypted applications, their performance on SSH classification is only 76% detection rate and 8% false positive rate. Bernaille et al. employed first clustering and then classification to the first few packets in each connection to identify SSL connections [11]. They use the first four packets of a TCP connection and represent it using 5-tuple (Destination/Source IP address, Destination/Source port numbers and Protocol) and the packet size. However, they have to wait for the connection to end to identify the first four packets. Even though this representation can enable early identification of an application, port numbers were used for classification in their work. Thus, not only its performance will drop when port numbers are changed but also it is not robust since it requires new training each time it is evaluated on a different network.

Another recent work by Williams et al. [5] compared five different classifiers namely, Bayesian Network, C4.5, Naïve Bayes (with discretisation and kernel density estimation) and Naïve 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. Unfortunately, this may be misleading especially on unbalanced data sets where, say, only 10% of the data set is in-class (SSH) and 90% out-class (non-SSH). Thus, by labeling everything as the major class, a classifier can achieve 90% accuracy. More recently, Alshammari et al. employed RIPPER and AdaBoost algorithms for classifying SSH traffic from offline log files without using any payload, IP addresses or port numbers [12]. In that work, public traces from MAWI and AMP repositories as well as testbed traces generated at authors' lab were employed. Results showed that RIPPER based classifier achieved 99% detection rate and 0.7% false positive rate at its best performance in the detection of SSH traffic.

On the other hand, Montigny-Leboeuf, from Communications Research Centre Canada (CRC), developed a number of indicators (attributes) that aim to portray essential communication dynamics based solely on information that can be gathered from monitoring packet headers in a traffic flow [27]. Based on these attributes rules are formed to classify different application traffic. A tool is developed at CRC to demonstrate the relevance of these rules and flow indicators in characterizing network traffic. Results reported on SSH traffic classification showed 79% detection rate and 5% false positive rate (and 13% of flows were unrecognized) [27]. These were achieved on a private data set at CRC where the applications are labelled using port numbers. Recently, it is shown that when such a heuristic based semi-automatic system is compared to a machine learning based fully automatic system, the machine learning based classification system outperformed the heuristics based one [29].

On the other hand, Skype analysis become very popular in the last few years, too. Baset et al present an analysis of the Skype behaviour [30]. Suh et al monitored Skype traffic using relay nodes [31]. Ehlert et al studied skype traffic to find signatures [32]. Bonfiglio et al adopted two techniques to detect skype traffic. These were Chi-Square test and Naive Bayesian classifier. However the results were evaluated with a payload-based classification scheme which gave the best results when the two detection methods combined [33]. Perenyi et al proposed Skype identification algorithm based on observable parts of Skype protocol. First candidate Skype host are detected using traditional IP and port-based identification together with a special signaling flow identification method. Then Skype calls are discovered exploiting the properties of speech flows, timing of voice packet and candidate hosts found in the first step [34]. Freire et al studied detecting skype flows in web traffic and achieved 100% detection rate with 5% false positive rate [35]. In contrast to the related work, our proposed system can potentially be applied to any encrypted application since it applies only flow based statistics without using the ip addresses, port numbers and payload data. Last but not the least, none of the previous work employing machine learning techniques to detect different application traffic investigated the robustness of such techniques to different encrypted applications and different network traces.

III. Methodology

As discussed earlier, in this work machine learning based classifiers are going to be employed in order to identify the most robust model/rules to the problem of SSH/Skype traffic classification. We believe that the robustness can be very important especially for forensic analysis where classifiers are naturally trained on a different network than the ones they are used to test/investigate.

A. Data Collection

In our experiments, the performance of the different machine learning algorithms is established on four different network data sources: Dalhousie traces, public traces (AMP and MAWI [13], [14]) and DARPA99 traces. The properties of the data sources are as follows:

- **Public traces** naturally have no reference to payload or network configuration. We used several public data sets from NLANR (National Laboratory for Applied Network Research - AMP data traces) [13] and MAWI (Measurement and Analysis on the WIDE Internet) [14] web sites. Brief statistics on the traffic data collected are given in Table I.
- **Dalhousie traces** were captured on the Dalhousie University Campus network by the University Computing and Information Services Centre (UCIS) in January 2007. Dalhousie is one of the biggest universities in the Atlantic region of Canada. There are more than 15000 students and 3300 faculty and staff. The UCIS is responsible for all the networking on the campus which includes more than 250 servers and 5000 computers. Moreover, the wireless network is enabled on the campus where thousands of users (students and staff) are connected daily. Dalhousie network is connected to the Internet via a full-duplex T1 fiber link. Full-duplex traffic on this connection was captured for 8 hours. Given the privacy related issues university may face, data is filtered to scramble the IP addresses and each packet is further truncated to the end of the IP header so that all payload is excluded. Moreover, the checksums are set to zero since they could conceivably leak information from short packets. However, any information regarding size of the packet is left intact. Brief statistics on the traffic data collected are given in Table I.
- **DARPA99** traces consists of five weeks of traces generated at the MIT Lincoln Labs for Intrusion Detection Evaluation [15]. The data represent a simulated network at an imaginary Air Force base. For each week, there are five network trace files that represent a network usage from 8:00 AM to 5:00 PM. We used data from week one and week three since these two weeks are attack-free and our purpose is to evaluate whether we can classify network traffic not if we can detect intrusions. We used only the inside sniffing data. Brief statistics on the traffic data collected are given in Table I.

For public data sets, there is no accurate method of determining the type of application to which different packets belong, since we do not have access to packet payloads. Thus, we simply use a port-based classifier to determine the class label, i.e. application, in these files, which use standard port numbers and label flows according to IANA assignments as done by the previous work [2], [4], [5], [16], [17], [10]. As for DARPA99 data sets, we again employed a port-based classifier to determine the class label. Indeed, this uses standard port numbers and labels flows according to IANA assignments as done by the previous work. However, given that DARPA99 traces also have payload, then we could verify the labels for SSH traffic against the handshake part of SSH in the payload (given that SSH handshake is not encrypted). This enabled us to know the ground truth for DARPA99 traces.

TABLE I
SUMMARY OF TRACES

	Dalhousie	**AMP**	**MAWI**	**DARPA99**
Total Packets	337041778	332,064,652	76543335	16723835
MBytes	213,562	188,435	28,718	3,638
% of TCP packets	86.51%	55.36%	85.37%	88.6%
% of TCP bytes	91.03%	72.05%	70.3%	93.29%
% of UDP packets	13.33%	33.6%	11.65%	11.33%
% of UDP bytes	8.95%	11.2%	5.5%	6.43%
% of Other packets	0.16%	11.04%	2.98%	0.07%
% of Other bytes	0.02%	16.75%	24.2%	0.28%

On the other hand, Dalhousie traces (UCIS) are labeled by UCIS by a commercial classification tool called PacketShaper, which is a deep packet analyzer [18]. PacketShaper uses Layer 7 filters (L7) to classify the applications [19].

Since all of the traffic traces are very large data sets. We performed subset sampling to limit the memory and CPU time required for training. In all cases, we sampled the training data sets from the Dalhousie traces since that represent the newest trace among the set. For SSH identification, the training data set, Dal Training Sample, is generated by sampling randomly selected (uniform probability) flows from five applications FTP, SSH, DNS, HTTP and MSN. In total, Dal Sample consists of 12246 flows, 6123 SSH and 6123 non-SSH. On the other hand, for Skype identification, Skype Dal Training Sample is generated by sampling randomly selected (uniform probability) flows from different classes (FTP, SSH, MAIL, DNS, HTTP, HTTPS and Random UDP). The applications in the “Random UDP” class includes random UDP 24000 flow instances. In total, Skype Dal Sample consists of 60000 balanced flows (skype flows vs non-skype flows).

B. Feature Selection

Network traffic is represented using flow-based features. In this case, each network flow is described by a set of statistical features. Here, a feature is a descriptive statistic that can be calculated from one or more packets. To this end, NetMate [20] is employed 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. The TCP flow time out value employed in this work is 600 seconds [21]. The flows as defined by the features, we extract a similar set of features

TABLE II

FLOW BASED FEATURES EMPLOYED

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

as in [5], form the input vector from which the machine learning model provides a label {SSH (Skye), non-SSH (non-Skype)} for each flow. As discussed earlier, features such as IP addresses, source/destination port numbers and payload are excluded from the feature set to ensure that the results are not dependent on such biased features.

C. Classifiers Employed

In order to identify SSH traffic; five different machine learning algorithms are deployed. These are Support Vector Machine (SVM), RIPPER, AdaBoost Naïve Bayesian and C4.5.

Support Vector Machines (SVMs) are a set of machine learning methods used for regression and classification problems [22]. They belong to a family of generalized linear classifiers. A special property of this family of classifiers is to simultaneously minimize the empirical classification error and maximize the geometric margin. Often, in classification problems, data points may not necessarily be points in $\Re^2$, but may be multidimensional $\Re^p$ or $\Re^n$ points. In this case, data is represented by a vector of *n* attributes or *n* features. The overall classification problem then takes the form of determining whether this data can be separated by a *n-1* dimensional hyperplane. Assuming our data is linearly separable; we should find a hyperplane that separates our feature vectors. This is a typical form of linear classifier. There are many linear classifiers that might satisfy this property. However, we are additionally interested in establishing the maximum separation/margin between the two classes. If such a hyperplane exists, the hyperplane is clearly of interest and is known as the maximum-margin hyperplane and such a linear classifier is known as a maximum margin classifier. The feature vectors from which the distance to the hyperplane is measured, or the vectors at either side of the margin, are known as the support vectors. A more detailed explanation of the algorithm can be found in [22].

AdaBoost, Adaptive Boosting, 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. 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 [23].

Naïve Bayesian is a statistical classifier based on Bayess theorem that gives its conditional probability a given class. This classification method analyses the relationship between instance of each class and each attributes to acquire a conditional probability for the relationships between the attribute values and the class. Naïve Bayesian classifier assumes the values of the input features are independent and have no effect on a given class. This assumption, conditional independence, is made to simplify the computations and consider to be naive. Naïve Bayesian can be used for classification in straightforward process by computing the probability of occurrence for each class , prior probability, and computing the probability of occurrence of instance in a given class. Moreover, Naïve Bayesian has managed to achieve good results even though when conditional independence assumption is violated. Further information on the Naïve Baysien algorithm can be found in [24].

C4.5 is a decision tree based classification algorithm. A decision tree is a hierarchical data structure for implementing a divide-and-conquer strategy. It is an efficient non-parametric method that can be used both for classification and regression. In non-parametric models, the input space is divided into local regions defined by a distance metric. In a decision tree, the local region is identified in a sequence of recursive splits in smaller number of steps. A decision tree is composed of internal decision nodes and terminal leaves. Each node *m* implements a test function *fm(x)* with discrete outcomes labeling the branches. This process starts at the root and is repeated until a leaf node is hit. The value of a leaf constitutes the output. In the case of a decision tree for classification, the goodness of a split is quantified by an impurity measure. A split is pure if for all branches, for all instances choosing a branch belongs to the same class after the split. A more detailed explanation of the algorithm can be found in [25].

RIPPER, Repeated Incremental Pruning to Produce Error Reduction, is a rule based machine learning algorithm. This means rules are learned from the data directly. Rule induction does a depth-first search and generates one rule at a time. Each rule is a conjunction of conditions on discrete or numeric attributes and these conditions are added one at a time to optimize some criterion. In RIPPER, conditions are added to the rule to maximize an information gain measure [26]. To measure the quality of a rule, minimum description length is used [26]. RIPPER stops adding rules when the description length of the rule base is 64 (or more) bits larger than the best description length. Once a rule is grown and pruned, it is added to the rule base and all the training examples that satisfy that rule are removed from the training set. Then the process continues until enough rules are added. In the algorithm, there is an outer loop in which one rule at a time is added to the rule base and an inner loop in which one condition at a time is added to the current rule. These steps are both greedy and do not guarantee optimality. A more detailed explanation of the algorithm can be found in [26].

IV. Experiments and Result

In traffic classification, two metrics are typically used in order to quantify the performance of the classifier: Detection Rate (DR) and False Positive Rate (FPR). In this case DR will reflect the number of SSH (Skype) flows correctly classified whereas FPR will reflect the number of non-SSH (non-Skype) flows incorrectly classified as SSH. Naturally, a high DR rate and a low FPR would be the desired outcomes. They are calculated as follows:

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

$$FPR = \frac{\#FPClassifications}{TotalNumberNon\_SSHClassifications}$$

where FN, False Negative, means SSH (Skype) traffic classified as non-SSH (non-Skype) traffic. Once the aforementioned feature vector is prepared for the data sets, then AdaBoost, Support Vector Machine, Naïve Bayesian, RIPPER and C4.5 based classifiers are trained on the Dal Training Samples. To this end, we have used Weka [28], which is an open source tool for data mining tasks. We employed Weka with its default parameters to run all algorithms on our data sets.

We use these trained models, on all of the complete traces employed. Table III lists the number of flows in each data set. Furthermore, Table I shows that the percentages of the TCP and UDP traffic are different for each trace. What this demonstrates is that these traces indeed belong to substantially different networks. Therefore, we believe that only well generalized models are able to classify SSH (Skype) traffic correctly on these networks.

Results show that the C4.5 and RIPPER based classifiers perform better than the other classifiers on the majority of the data sets. Our results show that C4.5 achieves the best results on Dalhousie, AMP and MAWI traces, whereas SVM achieves the best results on DARPA99 traces. Looking at only real network traces (DARPA99 is a simulated network trace), C4.5 performs better than the other classifiers. C4.5 achieves 95.9% DR and 2.8% FPR on Dalhousie traces, 97.2%, 97% DR and 0.8% FPR on the AMP traces, and 82.9% DR and 0.5% FPR on MAWI traces, Table IV. This not only shows that the model, which the C4.5 classifier learned during training, are adapted enough to be tested on real world network traces, but also verifies that accurate differentiation between SSH and non-SSH traffic is possible without employing port numbers, IP addresses and payload information.

TABLE III
Number of flows in the complete traces employed

	Dalhousie	AMP	MAWI	DARPA99
FTP	8504	14346	3395	8867
SSH	19384	427448	19016	72094
TELNET	510	4500	353	463643
MAIL	359212	174179	31410	173530
DNS	5325576	8021575	9601134	25735411
HTTP	5672886	450868	155511	474282
Skype	8664137	0	0	0
OTHERS	24179489	12004509	10163022	1633475
Total	44229698	21097425	19973841	28561302

Analysis of the model generated by C4.5 and RIPPER shows that C4.5 classifier model generates 13 rules for SSH flows and used 14 features, Figure 1 while RIPPER classifier model generates 11 rules for SSH flows and used 15 features, Figure 2. Both these models are easy to deploy and understandable by network administrators.

1) Results of Skype Experiments: Table V shows that C4.5 based classification approach is much better than other machine learning algorithms employed in identifying the Skype traffic. In this case, the classification based system can correctly classify ≈98% of the instances with less than 8% FPR.

V. Conclusion and Future Work

In this work, we investigate the robustness of the model/rules generated by AdaBoost, Support Vector Machine, Naïve Bayesian, RIPPER and C4.5 learning algorithms for distinguishing SSH traffic from non-SSH traffic in a given traffic trace. To do so, we employ public traffic traces from AMP and MAWI web sites based on the previous research [5] as well as employing traffic traces captured on our Dalhousie Campus network. We evaluated the aforementioned learning algorithms using traffic flow based features. We have employed Weka (with default settings) for both algorithms. Results show that the rules/model generated by C4.5 based classifier performs better than other based classifiers on the above data sets using flow based features. In these experiments, in the worst case scenario, the C4.5 based classifier can achieve a 83.7% DR and 1.5% FPR at its test performance (when trained on one network but tested on another) to detect SSH traffic. On the other hand, in the best case test scenario, C4.5 based classifier can achieve up to 97% DR and 0.8% FPR at its test performance. These results show that the classification based system trained on data from one network can be employed to run on a different network without new training. Thus, it can generalize well from one network data to another and is therefore robust. In short, the rules, i.e. solutions, generated by the classification based system are robust generic solutions as well as being easy to understand.

Last but not the least, we have also investigated classification based system on Dalhousie traces in order to

TABLE IV
RESULTS ON THE TRAINING DATA AND COMPLETE TRACES FOR SSH

	C4.5		**AdaBoost**		**Naive Bayesian**		**SVM**		**RIPPER**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Result of 10-fold cross validatoin on SSH Dal Training Sample										
Non-SSH	0.996	0.006	0.978	0.029	0.993	0.084	0.982	0.021	0.996	0.005
SSH	**0.994**	**0.004**	0.971	0.022	0.916	0.007	0.979	0.018	**0.995**	**0.004**
Result of Dalhousie traces										
Non-SSH	0.971	0.04	0.469	0.057	0.993	0.172	0.847	0.06	0.965	0.037
SSH	**0.959**	**0.028**	0.942	0.53	0.827	0.006	0.939	0.152	**0.962**	**0.034**
Result of AMP traces										
Non-SSH	0.991	0.027	0.372	0.963	0.994	0.998	0.709	0.011	0.99	0.061
SSH	**0.972**	**0.008**	0.036	0.627	0.001	0.005	0.988	0.29	0.938	0.009
Result of MAWI traces										
Non-SSH	0.994	0.17	0.207	0.761	0.998	0.985	0.523	0.087	0.994	0.184
SSH	**0.829**	**0.005**	0.238	0.792	0.014	0.001	0.912	0.476	**0.815**	**0.005**
Result of DARPA99 traces										
Non-SSH	0.988	0.166	0.958	0.105	0.97	0.309	0.962	0.001	0.973	0.111
SSH	**0.833**	**0.011**	0.894	0.041	0.69	0.029	**0.998**	**0.037**	**0.888**	**0.026**

TABLE V
RESULTS ON THE COMPLETE DALHOUSIE TRACES FOR SKYPE USING SKYPE DAL TRAINING SAMPLES

	C4.5		**AdaBoost**		**Naive Bayesian**		**SVM**		**RIPPER**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Result of 10-fold cross validatoin on Skype Dal Training Sample										
Non-SKYPE	0.981	0.022	0.876	0.114	0.525	0.046	0.276	0.042	0.974	0.033
SKYPE	**0.978**	**0.019**	0.886	0.124	0.954	0.475	0.958	0.724	0.968	0.026
Result of Skype Dal Training Sample on test data										
Non-SKYPE	0.923	0.016	0.821	0.134	0.401	0.079	0.45	0.11	0.938	0.03
SKYPE	**0.984**	**0.077**	0.866	0.179	0.921	0.6	0.89	0.55	0.971	0.062

identify Skype traffic. The preliminary results show that the classification based approach can achieve 98.4% detection rate and 7.8% false positive rate. Even though, the false positive rate in this case is high, these are promising results and should be investigated in more detail to develop a robust solution for Skype identification as well.

Future work will follow similar lines to compare the classification based approach against other clustering based approaches and to generate more data sets to test the robustness of the classifier for the classification of other encrypted applications, such as SSL and virtual private network tunnels. Moreover, the application of this approach to encryption algorithm identification will also be explored.

ACKNOWLEDGMENT

This work was in part supported by MITACS, NSERC and the CFI new opportunities program. Our thanks to John Sherwood, David Green and Dalhosuie UCIS team for providing us the anonymozied Dalhousie traffic traces. All research was conducted at the Dalhousie Faculty of Computer Science NIMS Laboratory, http://www.cs.dal.ca/projectx.

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] 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. 5-16 , 2006.

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

[7] 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.

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

```
max_bpktl <= 64
|  std_bpktl <= 5: NOTSSH (93.0)
|  std_bpktl > 5
|  |  total_fvolume <= 390: SSH (5664.0)
|  |  total_fvolume > 390
|  |  |  total_fpackets <= 4: NOTSSH (3.0)
|  |  |  total_fpackets > 4: SSH (23.0)
max_bpktl > 64
|  max_biat <= 673862
|  |  min_fpktl <= 90
|  |  |  max_fiat <= 1231899
|  |  |  |  total_fpackets <= 10: NOTSSH (4558.0/11.0)
|  |  |  |  total_fpackets > 10
|  |  |  |  |  total_bvolume <= 4116
|  |  |  |  |  |  max_bpktl <= 454: NOTSSH (29.0)
|  |  |  |  |  |  max_bpktl > 454
|  |  |  |  |  |  |  mean_fpktl <= 231: SSH (30.0)
|  |  |  |  |  |  |  mean_fpktl > 231: NOTSSH (7.0)
|  |  |  |  |  total_bvolume > 4116: NOTSSH (535.0/1.0)
|  |  |  max_fiat > 1231899
|  |  |  |  max_fiat <= 1446690: NOTSSH (14.0)
|  |  |  |  max_fiat > 1446690: SSH (11.0)
|  |  min_fpktl > 90
|  |  |  max_bpktl <= 236: NOTSSH (62.0)
|  |  |  max_bpktl > 236
|  |  |  |  min_fpktl <= 93
|  |  |  |  |  min_fpktl <= 92: SSH (43.0)
|  |  |  |  |  min_fpktl > 92
|  |  |  |  |  |  min_bpktl <= 263: SSH (8.0)
|  |  |  |  |  |  min_bpktl > 263: NOTSSH (2.0)
|  |  |  |  min_fpktl > 93: NOTSSH (6.0)
|  max_biat > 673862
|  |  mean_bpktl <= 286
|  |  |  proto <= 6
|  |  |  |  std_fiat <= 425439
|  |  |  |  |  std_fpktl <= 224
|  |  |  |  |  |  max_fpktl <= 535
|  |  |  |  |  |  |  max_fpktl <= 89: SSH (81.0)
|  |  |  |  |  |  |  max_fpktl > 89
|  |  |  |  |  |  |  |  max_bpktl <= 577: NOTSSH (30.0)
|  |  |  |  |  |  |  |  max_bpktl > 577
|  |  |  |  |  |  |  |  |  total_fpackets <= 6: NOTSSH (2.0)
|  |  |  |  |  |  |  |  |  total_fpackets > 6: SSH (15.0)
|  |  |  |  |  |  max_fpktl > 535: SSH (189.0)
|  |  |  |  |  std_fpktl > 224
|  |  |  |  |  |  total_fvolume <= 41312
|  |  |  |  |  |  |  min_fpktl <= 45: NOTSSH (92.0/3.0)
|  |  |  |  |  |  |  min_fpktl > 45
|  |  |  |  |  |  |  |  max_bpktl <= 873: SSH (4.0)
|  |  |  |  |  |  |  |  max_bpktl > 873: NOTSSH (12.0)
|  |  |  |  |  |  total_fvolume > 41312: SSH (27.0)
|  |  |  |  std_fiat > 425439: NOTSSH (84.0/3.0)
|  |  |  proto > 6: NOTSSH (137.0)
|  |  mean_bpktl > 286
|  |  |  std_fpktl <= 161
|  |  |  |  total_fvolume <= 3106: NOTSSH (78.0)
|  |  |  |  total_fvolume > 3106
|  |  |  |  |  mean_bpktl <= 823: SSH (13.0)
|  |  |  |  |  mean_bpktl > 823
|  |  |  |  |  |  min_fpktl <= 45: NOTSSH (30.0)
|  |  |  |  |  |  min_fpktl > 45
|  |  |  |  |  |  |  max_fpktl <= 625: NOTSSH (3.0)
|  |  |  |  |  |  |  max_fpktl > 625: SSH (5.0)
|  |  |  std_fpktl > 161: NOTSSH (372.0)
```

Fig. 1. C4.5 Model

```
(max_bpktl <= 64) and (std_bpktl >= 8) => class=SSH (5631.0/1.0)
(max_biat >= 674139) and (mean_bpktl <= 240) and (max_bpktl >= 576) and (std_fpktl <= 254) and
(total_fpackets >= 7) => class=SSH (196.0/0.0)
(max_biat >= 804275) and (mean_bpktl <= 100) and (max_fpktl <= 83) => class=SSH (137.0/0.0)
(min_fpktl >= 91) and (min_bpktl >= 238) and (min_fpktl <= 91) => class=SSH (43.0/0.0)
(total_fpackets >= 10) and (mean_bpktl <= 447) and (max_bpktl >= 584) and (mean_fpktl <= 160) =>
class=SSH (57.0/1.0)
(total_fvolume >= 27500) and (total_fvolume >= 78256) => class=SSH (23.0/0.0)
(max_biat >= 951987) and (max_fiat >= 1655372) and (mean_fpktl <= 90) => class=SSH (8.0/0.0)
(min_fpktl >= 93) and (mean_fpktl <= 93) and (duration >= 500326) => class=SSH (8.0/0.0)
(min_fiat >= 4108) and (total_fvolume >= 7408) and (min_fpktl >= 52) => class=SSH (5.0/0.0)
(min_biat >= 189) and (max_bpktl >= 584) and (std_bpktl <= 331) and (mean_fpktl <= 201) and
(mean_fpktl >= 139) and (max_bpktl >= 680) => class=SSH (9.0/0.0)
(total_fvolume >= 5668) and (max_bpktl <= 696) and (max_bpktl >= 680) => class=SSH (5.0/0.0)
 => class=NOTSSH (6140.0/11.0)
```

Fig. 2. RIPPER Model

[9] Karagiannis, T., Papagiannaki, K., and Faloutsos, M, “BLINC: Multilevel Traffic Classification in the Dark”,Proceedings of Applications, Technologies, Architectures, and Protocols For Computer Communications pp 229-240, 2005.

[10] 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.

[11] Bernaille L., Teixeira R., “Early Recognition of Encrypted Applications”, Passive and Active Measurement Conference (PAM), Louvain-la-neuve, Belgium, April, 2007.

[12] Alshammari, Riyad; Nur Zincir-Heywood, A., “A flow based approach for SSH traffic detection,” Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on , vol., no., pp.296-301, 7-10 Oct, 2007.

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

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

[15] 1999 DARPA intrusion detection evaluation data, http://www.ll.mit.edu/IST/ideval/docs/1999schedule.html, last accessed Jan. 2008.

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

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

[18] PacketShaper, http://www.packeteer.com/products/packetshaper/.

[19] l7-filter, http://l7-filter.sourceforge.net/

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

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

[22] Burges C. J. C., “A Tutorial on Support Vector Machines for Pattern Recognition”, Data Mining and Knowledge Discovery, 2(2): 1-47, 1998.

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

[24] George H. John and Pat Langley Estimating Continuous Distributions in Bayesian Classifiers. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. pp. 338-345, Morgan Kaufmann, San Mateo, 1995.

[25] J R Quinlan, “C4.5: Programs for Machine Learning”,Morgan Kaufmann Publishers,isbn=1-55860-238-0, 1993.

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

[27] Montigny-Leboeuf A., Flow Attributes For Use In Traffic Characterization, Journal of CRC Technical Note No. CRC-TN-2005-003, Ottawa, ON, Canada, 2005.

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

[29] Alshammari, Riyad; Zincir-Heywood, A. Nur, “Investigating Two Different Approaches for Encrypted Traffic Classification, ” Privacy, Security and Trust, 2008. PST ’08. Sixth Annual Conference on , vol., no., pp.156-166, 1-3 Oct. 2008

[30] S. Baset, H. Schulzrine, “An analysis of the skype peer-to-peer internet telephony protocol,” in INFOCOM06: Proceedings of the 25th IEEE International Conference on Computer Communications, 2006.

[31] K Suh, D. R. Figueiredo, J. Kurose, and D. Towsley, “Characterizing and detecting relayed traffic: A case study using skype,” in INFOCOM 06: Proceedings of the 25th IEEE International Conference on Computer Communications, Apr 2006.

[32] S. Ehlert, S. Petgang, T. Magedanz, and D. Sisalem, “Analysis and signature of skype VoIP session traffic,” in CIIT 2006: 4th IASTED

International Conference on Communications, Internet, and Information Technology, Nov/Dec 2006, pp. 8389.

[33] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, “Detailed analysis of skype traffic”, IEEE Transactions on Multimedia, Vol. 11, No.1, Jan 2009.

[34] M. Perenyi, A. Gefferth, T. D. Drang, S. Molnar, “Skype traffic identification”, IEEE, 2007.

[35] E. P. Freire, A. Ziviani, R, M. Salles, “Detecting Skype flows in web traffic”, IEEE, 2008.