Classifying SSH Encrypted Traffic with Minimum Packet Header Features using Genetic Programming

Riyad Alshammari, Peter Lichodzijewski, Malcolm I. Heywood, A. Nur Zincir-Heywood
Faculty of Computer Science
Dalhousie University
Halifax, NS, Canada
{riyad,piotr,mheywood,zincir}@cs.dal.ca

ABSTRACT

The classification of Encrypted Traffic, namely Secure Shell (SSH), on the fly from network TCP traffic represents a particularly challenging application domain for machine learning. Solutions should ideally be both simple – therefore efficient to deploy – and accurate. Recent advances to team-based Genetic Programming provide the opportunity to decompose the original problem into a subset of classifiers with non-overlapping behaviors, in effect providing further insight into the problem domain and increasing the throughput of solutions. Thus, in this work we have investigated the identification of SSH encrypted traffic based on packet header features without using IP addresses, port numbers and payload data. Evaluation of C4.5 and AdaBoost – representing current best practice – against the Symbiotic Bid-based (SBB) paradigm of team-based Genetic Programming (GP) under data sets common and independent from the training condition indicates that SBB based GP solutions are capable of providing simpler solutions without sacrificing accuracy.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Performance, Algorithms

Keywords

Genetic Programming, supervised learning, coevolution, encrypted traffic classification, packet, active learning, efficiency, problem decomposition, teaming, and security

1. INTRODUCTION

The accurate classification of network traffic according to application type represents a challenging decision making activity that is not only important for network management but also important for defense applications since it can facilitate the assessment of security threats. Such a classification system is particularly useful from a defense application perspective since most of the time users with malicious intents would try to hide their traffic either in encrypted or covert channels. Thus, systems that can classify encrypted traffic represent a first step in identifying such malicious users. Moreover such systems can also be useful as a forensic tool to identify applications used by malicious users whose data is collected/captured by law-enforcement units. In this case, establishing a classification of traffic types in real-time can actually reflect the current utilization of applications and services in a given traffic trace. This in return can help law-enforcement units to predict the true intent of malicious users. Naturally, the process of traffic classification has several unique challenges including: non-standard utilization of ports, embedding of services within encrypted channels, dynamic port-to-application relationships, and the real-time nature of the domain. Such factors increasingly preclude the utilization of hand crafted rules or deep packet analysis. The former are likely to fail when the assumptions behind the rules no longer hold (as in assumptions regarding the mapping between application-to-port) and the latter represent an overhead in complexity under real-time contexts.

The basic approach of this work is therefore to assume a machine learning approach for discovering appropriate rules to distinguish between encrypted and non-encrypted traffic – specifically Secure Shell (SSH) versus non-SSH – *without* access to Internet Protocol (IP) addresses, port numbers and payload data. This approach immediately raises several fundamental questions, including: how to establish the data on which 'general' – as opposed to network specific – classification rules are identified, what representation and corresponding features to assume, and what representation should the model of classification assume to satisfy both real-time and accuracy requirements. In this work, use is made of training and test data from entirely independent networks in order to provide some measure of classifier generalization. Issues of data representation are addressed by employing packet header based features only without using IP addresses, port numbers and payload data. Specifically, a representation based on packet header information implies a low overhead when deriving corresponding features in real time, whereas the construction of statistical flows is a much more involved process[1].

[1]Flows are derived from a 5-tuple consisting of protocol (TCP), 'forward' and 'backward' IP addresses and corre-

From an accuracy perspective, inspecting the payload of every packet would provide the optimal detection performance, especially when the payload is not encrypted; however, encrypted payload makes this irrelevant. At the other extreme, network managers might rely on Transmission Control Protocol (TCP)/UDP User Datagram Protocol (UDP) port numbers for classification purposes. However this approach has become increasingly inaccurate, mostly because applications use non-standard ports to avoid detection, to bypass firewalls, or circumvent operating systems restrictions. Moreover, ports can be dynamically allocated as needed e.g., the same port number can be used to transmit multiple applications, most notably port 80. In short, other techniques are needed to increase the accuracy of network traffic identification.

One possibility is to identify specific features of the application traffic and use these features to guide traffic classification. Recent research in this area focuses on the identification of efficient models of classification. Different research groups have employed various classification techniques such as Hidden Markov and Naïve Bayesian models, AdaBoost, or Maximum Entropy techniques to this problem [9, 16, 15, 24]. On the other hand, the limitations of port- and/or payload-based analysis (which operate at the packet level) have motivated the use of transport/flow layer statistics for traffic classification [5, 8, 10]. These techniques assume that different applications have distinct behavioral patters on the network. However, in general all these efforts show that even though it is easier to apply such techniques to well known public domain applications such as Web (WWW) and mail, more work is needed to distinguish between Peer-to-Peer (P2P) or SSH type of encrypted applications accurately. Moreover, P2P and encrypted applications such as Skype not only use the port numbers dynamically, but also use the same port number for multiple applications.

In this work, we focus on SSH traffic as a case study of encrypted traffic identification with the following rationale: (i) SSH is an encrypted application but also public domain, hence we know how the protocol works and therefore can know the ground-truth (correct labels) in the data sets employed; (ii) Even though SSH is an encrypted application, the handshake part of the SSH protocol is actually not encrypted and this further reinforces ground-truth of the data set. Needless to say, none of this would have been the case if we had studied an application such as Skype in the case study, making it much more difficult to provide definitive statements on classifier performance. Moreover, 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 and multiple applications can be run within an SSH session, thus potentially changing the behavioral profile (from this perspective, it is similar to Skype).

The objective of this work is to examine the utility of team-based GP against the two most frequently preferred machine learning approaches for network traffic classification, specifically C4.5 and AdaBoost. All three methods support the embedded paradigm of building classifiers – that is to say, attribute selection is performed as an integral part of learning. Conversely, assuming a team-based model of GP provides the opportunity to further decompose the classification problem into an arbitrary number of independent models. This potentially provides a solution in terms of simpler models than would be the case when assuming a single binary classifier per class, or compared to ensemble approaches in which significant overlap in classifier behavior renders the solution opaque [14]. The specific form of team-based GP employed here takes the form of the Symbiotic Bid-based (SBB) paradigm[2] where this is known to be computationally efficient (care of a competitive coevolutionary formulation of active learning) and does not require the a priori specification of the number of team members (care of a bid-based model of cooperation) [13].

sponding port numbers. When IP numbers match within a finite temporal window 'flow' statistics are calculated.

2. RELATED WORK

SSH is an application, which enables a user to log into another computer over a network, to execute commands on the remote machine, and to move files from one machine to another. SSH provides strong authentication and secure communications over unsecured channels. Moreover, it provides secure X connections and secure forwarding of arbitrary TCP connections [20]. It is intended as a replacement for rlogin, rsh and rcp where these commands are vulnerable to different kinds of attacks. Examples include IP spoofing, IP source routing, Domain Name System (DNS) spoofing, interception of clear text passwords and other data, manipulation of data and attacks based on listening to X authentication data and spoofed connection to the X11 server. To avoid these attacks, SSH encrypts all communications. As discussed earlier, having an encrypted payload and being able to run different applications over SSH makes it a challenging problem to detect SSH traffic from a given traffic log file. Support for encryption implies that performing payload analysis is not useful. Moreover port number based SSH classification cannot be accurate either given that non-standard ports can be used to run SSH in order to avoid detection or bypass firewalls. Most of existing research therefore focuses on automatic application recognition in general i.e., classification of public domain applications such as WWW, mail, or file transfer protocol (FTP); and therefore does not address issues pertinent to the classification of encrypted traffic.

Zhang and Paxson present one of the earliest studies of techniques based on matching patterns in the packet payload [26]. Moore *et al.* used Bayesian analysis to classify flows into broad categories such as bulk transfer, P2P or interactive [15, 16]. Haffner *et al.* compared AdaBoost, Hidden Markov, Naïve Bayesian and Maximum Entropy models to classify network traffic into different applications [9]; recommending AdaBoost as the most appropriate, with SSH classified at an 86% detection rate and 0% false positive rate. However, to do so they employed the first 64 bytes of the payload. Given that most SSH applications use a specific encryption algorithm, this factor increases the correct detection of SSH. However, if the encryption algorithm is changed, the effectiveness of such a system might be expected to decrease. Specifically, this approach is not generic enough to apply against other encrypted applications such

[2]Source code available from http://www.cs.dal.ca/~mheywood/Code/SBB/SCM.9.r20081212.tar.gz

as Skype or Virtual private network tunnels. Karagiannis *et al.* proposed an approach that does not use port numbers or payload information [10]. Instead this approach relies on information about the behavior of the hosts on the network. Specifically, they make use of social and functional roles of hosts such as employing, *a priori* knowledge regarding the interaction with other hosts i.e., whether a host acts as a provider or a consumer of a service. Such an approach is 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. More recently, Wright *et al.* investigated the extent to which common application protocols can be identified using only packet size, timing and direction information of connection information [24, 25]. They compared k-Nearest Neighbor (kNN) and Hidden Markov Model classifiers. The resulting evaluation indicated that distinct encrypted applications could be detected, but performance on SSH classification (76% detection rate and 8% false positive rate) was not as good as public domain applications such as WWW and instant messaging. Another recent work by Williams *et al.* [22] compared five different classifiers namely, Bayesian Network, C4.5, Naïve Bayes (with discretisation and kernel density estimation) and a Naïve Bayes Tree, on the task of traffic flow classification. They found that C4.5 was the most reliable model. However, rather than giving classification results per application, they give overall accuracy per machine learning algorithm. Unfortunately, this may be misleading especially on unbalanced data sets where say only 5% of the data set is in-class and 95% out-class. Alshammari *et al.* investigated the performance of C4.5 and RIPPER algorithms on three different feature sets (flow, heuristic and packet header features) for the classification of encrypted traffic such as SSH, but without using features such as IP addresses, source/destination ports and payload information [3, 4]. Results indicated that C4.5 provides the highest performance using flow based feature sets. Finally, we note that none of the previous works employing machine learning techniques to detect different application traffic investigated the robustness of the resulting solutions on test data derived from entirely different networks.

3. CLASSIFIER METHODOLOGIES

Given the general success of C4.5 and AdaBoost in previous studies, we employ both models during this study in order to establish a performance baseline. A more detailed explanation of these algorithms can be found in [2]. The following will summarize both schemes before introducing the SBB model of team-based GP.

3.1 C4.5

C4.5 is a decision tree based classification algorithm. A decision tree is a hierarchical data structure for implementing a divide-and-conquer strategy of attribute based model building. It is an efficient non-parametric method applicable both to classification and regression. Non-parametric models divide the input space 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 encountered. 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, typically entropy based. Naturally, if the split is not 'pure', then the instances should be split to decrease impurity, and there are multiple possible attributes on which a split can be performed. Such a scheme is locally optimal, hence has no guarantee on finding the smallest decision tree.

3.2 AdaBoost

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.

3.3 Team-based Genetic Programming

3.3.1 Overview

The canonical framework for applying model based cases of evolution – such as GP – to the supervised learning domain of classification requires an individual to map exemplars from an attribute space to a class label space. An individual's program expresses the mapping. However, this is not the case under the SBB framework [12]. Instead the task is divided into two components: (1) deciding *which* exemplars to label, or the *bid*, and (2) suggesting class label, or the *action*. In the case of the individual's action, the assumption is made that an individual will always be associated with the same action (class label). Thus at initialization, a problem with C classes results in $\frac{PopSize}{C}$ of the individuals in the population being pre-assigned to each class. The assignment is defined by assigning a scalar a to each individual at initialization. Scalars are selected with uniform probability from the set $\{1, ..., C\}$. The actions are *not adapted* during evolution. Conversely, the task of deciding which subset of exemplars to label is expressed in terms of a bid. The individual with the maximum (winning) bid suggests its pre-assigned action as the class label. During training, individuals are rewarded for bidding high only on exemplars whose class label matches their action.

The most recent form of the bid-based framework makes extensive use of coevolution [13], with a total of three populations involved: a population of points, a population of learners, and a population of teams (Figure 1). Specifically, individuals comprising a team are specified by the team population, thus establishing a symbiotic relationship with the learner population. Only the subset of individuals indexed by an individual in the team population compete to bid against each other on training exemplars. The use of a symbiotic relation between teams and learners makes the credit assignment process more transparent than in the

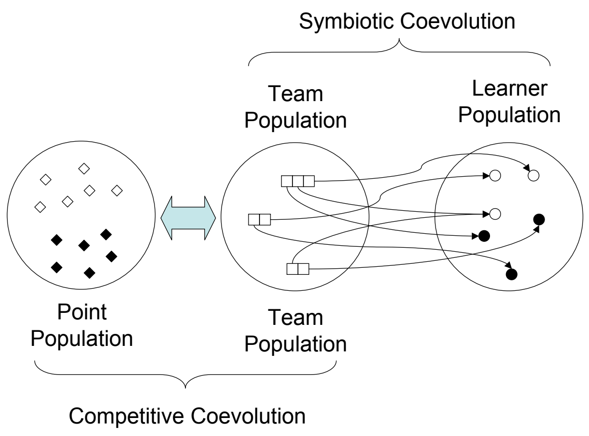

Figure 1: Architecture of Symbolic Bid-based GP: Point to team populations are competitive, Team to learner populations are symbiotic (cooperative).

case of a population wide competition between bids (as used in the earlier variant of the model [12]). Thus, variation operators may now be defined for independently investigating team composition (team population) and bidding strategy (learner population). The third population provides the mechanism for scaling evolution to large data sets. In particular the interaction between team and point population is formulated in terms of a competitive coevolutionary relation [6]. As such, the point population indexes a subset of the training data set under an active learning model (i.e. the subset indexed varies as classifier performance improves). Biases are enforced to ensure equal sampling of each class, irrespective of their original exemplar class distribution [7], whereas the concept of Pareto competitive coevolution is used to retain points of most relevance to the competitive coevolution of teams.

3.3.2 *SBB Algorithm*

The SBB model of evolution generates P_{gap} new exemplar indexes in the point population and M_{gap} new teams in the team population at each generation. Specifically, individuals in the point population take the form of indexes to the training data and are generated stochastically (subject to the aforementioned class balancing heuristic). New teams are created through variation operators applied to the current team population. Fitness evaluation evaluates all teams against all points with $(P_{size} - P_{gap})$ points and $(M_{size} - M_{gap})$ teams appearing in the next generation. Pareto competitive coevolution ranks the performance of teams in terms of a vector of outcomes, thus the Pareto non-dominated teams are ranked the highest [6]. Likewise, the points supporting the identification of non-dominated individuals (distinctions) are also retained. In addition, use is made of competitive fitness sharing in order to bias survival in favor of teams that exhibit uniqueness in the non-dominated set (Pareto front).

Evaluation of team m_i on a training exemplar defined by point population member p_k results in the construction of an outcome matrix $G(m_i, p_k)$ in which unity implies a correctly classified exemplar, and zero an incorrectly classified exemplar. A distinction for point p_k is defined as

$$\begin{cases} 1 & \text{if } G(m_i, p_k) > G(m_j, p_k) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where unity implies that point p_k 'distinguishes' between team m_i and m_j, and the result of all such pairwise team comparisons defines the objectives for the point. The ensuing Pareto competitive coevolutionary process identifies the non-dominated teams and points supporting their identification.

Denoting the non-dominated and dominated points as $F(P)$ and $D(P)$ respectively, the SBB framework notes that as long as $F(P)$ contains less than $(P_{size} - P_{gap})$ points, all the points from $F(P)$ are copied into the next generation. On the other hand, if $F(P)$ contains more points than are allowed to survive, then the following fitness sharing heuristic is imposed to rank the collection of non-dominated points [18],

$$\sum_i \frac{d_k[i]}{1 + N_i} \tag{2}$$

where $d_k[i]$ is the ith entry of the distinction vector for p_k; and N_i is the sum of the i th entries over the distinction vectors across all points in $F(P)$ i.e., the number of points making the same distinction. Thus, points making the same distinction are weighted less than points making unique distinctions.

An analogous process is repeated for the case of team selection, with $(M_{size} - M_{gap})$ individuals copied into the next generation. Naturally, under the condition where the (team) non-dominated set exceeds this number, the fitness sharing ranking employs $F(M)$ and $D(M)$ in place of $F(P)$ and $D(P)$ respectively. The resulting process of fitness sharing under a Pareto model of competitive coevolution has been shown to be effective at promoting solutions in which multiple models cooperate to decompose the original $|C|$ class problem into a set of non-overlapping behaviors [12], [13].

Finally, the learner population of individuals expressing specific bidding strategies employs a linear representation. Bid values are standardized to the unit interval through the use of a sigmoid function, or $bid(y) = (1 + \exp -y)^{-1}$, where y is the real valued result of program execution on the current exemplar. Variation operators take the form of instruction add, delete, swap and mutate, applied with independent likelihoods, under a uniform probability of selection. When an individual is no longer indexed by the team population it becomes extinct and deleted from the learner population. Conversely, during evaluation of the team population, exactly M_{gap} children are created pairwise care of team based crossover. Learners that are common to both child teams are considered to be the candidates for retention. Learners not common to the child teams are subject to stochastic deletion or modification, with corresponding tests for deletion/ insertion at the learner population. The instruction set follows from that assumed in [13] and consists of eight opcodes $(\{cos, exp, log, +, \times, -, \div, \%\})$ operating on up to 8 registers, as per a linear GP representation.

4. EVALUATION METHODOLOGY

As discussed earlier, the preferred models of classifications from Section 2 (C4.5 and AdaBoost) will be compared against SBB based GP under the packet based SSH traffic

Table 1: Summary of traces packet count and application types

	University	**DARPA99**
Total Packets	337041778	16723835
MBytes	213,562	3,638
% of TCP packets	86.51%	88.6%
% of TCP bytes	91.03%	93.29%
% of UDP packets	13.33%	11.33%
% of UDP bytes	8.95%	6.43%
% of Other packets	0.16%	0.07%
% of Other bytes	0.02%	0.28%

classification problem. Solution robustness will be assessed by training on a data set from one location (hereafter denoted University) but tested on a data set from another location (hereafter DARPA1999). The properties of the data sources are as follows:

- **University data sets** were captured by the Dalhousie University Computing and Information Services Centre (UCIS) in January 2007 on the Dalhousie campus network between the university and the commercial Internet. Given the privacy related issues universities face, data is filtered to scramble the IP addresses and further truncate each packet to the end of the IP header so that all the payload is excluded. Moreover, the checksums are set to zero since they could conceivably leak information from short packets. However, any length information in the packet is left intact. Summary statistics on the traffic data collected are given in Table 1.

- **DARPA99** traces consist of five weeks of traces generated at the MIT Lincoln Labs for Intrusion Detection Evaluation [1]. For each week, there are five network trace files that represent a network usage from 8:00 AM to 5:00 PM. Data was used from weeks 1 and 3 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. Only the 'inside' sniffing data was employed. Summary statistics on the traffic data collected are given in Table 1.

The University traces are labeled by a commercial classification tool (PacketShaper), which is a deep packet analyzer [17]. PacketShaper uses Layer 7 filters (L7) to classify applications [11]. Thus, by deep packet inspection, the handshake part of the SSH protocol can easily be identified since that part is not encrypted. We can therefore assume with a high degree of confidence that the labeling of the data set is 100% correct and provides us the ground-truth for the University traces.

On the other hand, a port-based classifier was employed for labeling the DARPA99 traces. This assumes IANA assignments and follows the approach taken in previous work [5, 8, 9, 15, 22, 25]. Given that DARPA99 traces also have payload, the labels for SSH traffic were additional verified against the handshake part of SSH in the payload. This enabled us to establish the ground-truth for DARPA99 traces, as well.

Naturally, both of the traffic traces represent large data sets from a machine learning perspective, thus subset sampling is used to decouple the overall exemplar count from the

Table 2: UniversitySamples - # of packets

SSH	250000	FTP	30000
FTPDATA	30000	TELNET	10000
MAIL	30000	DNS	30000
HTTP	30000	HTTPS	30000
IMAPS	30000	POP3S	30000

subset over which training is conducted. Indeed when subsampling algorithms impose a simple class balance heuristic, performance of the resulting models is frequently better than when training over all exemplars. Specifically, the balanced subset sampling heuristic results in performance correlated with maximization of the AUC (Area Under the Receiver Operating characteristic (ROC) Curve) statistic [21].

In this work, we have used a sampled subset of University traces as training and the rest of the University traces and DARPA99 traces (weeks 1 and 3) as testing. The training data set is generated by sampling randomly selected (uniform probability) packets from ten applications, Table 2. In total, the sample consists of 500,000 packets, of which 250,000 are SSH and 250,000 are non-SSH.

Each packet is described in terms of 39 attributes (Table 3), where the underlying principle will be that features employed are simple and clearly defined within the networking community. They represent a reasonable benchmark feature set to which more complex features might be added in the future. To this end, Wireshark [23] is employed to process data sets and generate feature values. As discussed earlier, we did not use the IP addresses, port numbers and payload data as this generally results in solutions that are specific to a particular network sample as opposed to generalizing to solutions of wider utility.

5. EMPIRICAL EVALUATION

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). In this case DR will reflect the number of SSH flows correctly classified, whereas FP rate will reflect the number of Non-SSH flows incorrectly classified as SSH. Naturally, a high DR rate and a low FP rate are the most desirable outcomes. They are defined as follows:

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

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

where FN, False Negative, implies that SSH traffic is classified as non-SSH traffic.

All three candidate classifiers are trained on the training data using a 10-fold cross validation. Weka [19] with its default parameters is used to build C4.5, and AdaBoost solutions. The SBB based GP classifier default parameters are summarized in Table 4. Thirty runs of the SBB algorithm were performed using different population initializations. Figure 2 summarizes solutions with a (training) DR higher than 50%; there are two that are non-dominated. These two solutions are then evaluated on the test data sets and compared against C4.5 and AdaBoost solutions.

Table 3: Packet Header based features employed, * Normalized by log

frame.time delta	frame.pkt_len*	frame.len*	frame.cap_len*
frame.marked	ip.len*	ip.flags	ip.flags.rb
ip.flags.df	ip.flags.mf	ip.frag.offset	ip.ttl*
ip.proto	ip.checksum	ip.checksum_good	ip.checksum_bad
tcp.len*	tcp.seq*	tcp.nxtseq*	tcp.ack*
tcp.hdr_len*	tcp.flags*	tcp.flags.cwr	tcp.flags.ecn
tcp.flags.urg	tcp.flags.ack	tcp.flags.push	tcp.flags.reset
tcp.flags.syn	tcp.flags.fin	tcp.window_size*	tcp.checksum
tcp.checksum_good	tcp.checksum_bad	udp.length*	udp.checksum_coverage*
udp.checksum	udp.checksum_good	udp.checksum_bad	

Table 4: SBB model algorithm parameterization.

	Description	Value
P_{size}	Point population size.	90
M_{size}	Team population size.	90
t_{max}	Number of generations.	30000
p_d	Probability of learner deletion.	0.1
p_a	Probability of learner addition.	0.2
μ_a	Probability of learner mutation.	0.1
ω	Maximum team size.	30
P_{gap}	Point generation gap.	30
M_{gap}	Team generation gap.	60

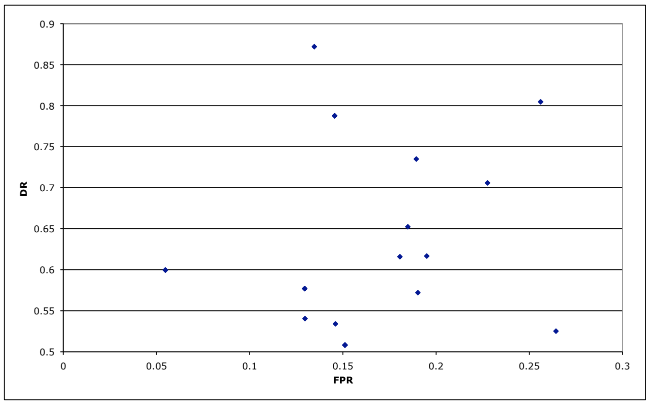

Figure 2: Scatter plot for the GP training performance (DR versus FPR).

Table 5: Results on Data sets.

	C4.5		AdaBoost		GP	
	DR	FPR	DR	FPR	DR	FPR
Training Sample						
Non-SSH	0.995	0.005	0.828	0.252	0.865	0.13
SSH	0.995	0.005	0.748	0.172	0.87	0.13
University Traces						
Non-SSH	0.930	0.087	0.839	0.204	0.803	0.114
SSH	0.913	0.07	0.796	0.161	0.886	0.197
DARPA99 Traces						
Non-SSH	0.791	0.885	0.727	0.023	0.866	0.274
SSH	0.115	0.209	0.977	0.273	0.726	0.134

Table 6: Features used by AdaBoost as a whole

1	frame.cap_len	frame.cap_len
2	tcp.ack	tcp.ack
3	tcp.seq	tcp.seq
4	tcp.window_size	tcp.window_size
5	tcp.len	tcp.len
6	ip.ttl	ip.ttl
7	tcp.flags.reset	tcp.flags.reset
8	ud.plength	udp.length
9	tcp.nxtseq	tcp.nxtseq

5.1 Results

Results are summarized in terms of both accuracy and model complexity. Table 5 lists the results for the three machine learning algorithms on the (University) training and 'test (validation)' traces, and independent test (DARPA99) traces. That is to say, all model construction takes place on the University Training partition. Post training evaluation is conducted under University and DARPA99 traces where neither were encountered during training. Naturally, the University Test partition will more closely reflect the training behavior than the DARPA99 test partition. All models tend to return a marginally better detection rate on out-class data than in-class data, whereas the false positive rates are generally better under the in-class data. Moreover, under the training partition, C4.5 appears to provide the stronger performance with consistently better in- and out-class FP and DR under both University data training and validation partitions. Conversely, AdaBoost was the weaker classifier of the three under the University data.

Introducing the entirely independent test set – DARPA99 – indicated that C4.5 had over-learnt the properties implicit in the training partition. Indeed most out-class data was classified as SSH. Moreover, AdaBoost was observed to provide best case performance under the independent test partition. The SBB based GP classifier was the most consistent performer across all test and training conditions, with results under the University partitions clearly better than AdaBoost, while also being competitive with AdaBoost under DARPA99 (particularly in terms of SSH False Positive rate and non-SHH Detection rate).

In all cases, the approach adopted to attribute selection was to include as wide a set as possible and let the 'embedded' properties of the various learning algorithms establish which subset of attributes to actually employ. Given this capability, we are now in a position to review the attributes selected by each model, where this is readily achieved class-

Table 7: Features used by GP for in/out-class

	in-class	out-class
1	tcp.seq	tcp.seq
2	tcp.ack	tcp.ack
3	tcp.flags	tcp.flags
4	ip.flags.df	ip.flags
5		ip.ttl
6		ip.checksum
7		ip.checksum_bad
8		tcp.hd_len
9		tcp.flags.urg
10		tcp.flags.reset
11		tcpwindow_size

Table 8: Features used by C4.5 for in/out-class

	in-class	out-class
1	frame.cap_len	frame.cap_len
2	tcp.ack	tcp.ack
3	ip.ttl	ip.ttl
4	tcp.window_size	tcp.window_size
5	tcp.seq	tcp.seq
6	tcp.nxtseq	tcp.nxtseq
7	frame.pkt_len	frame.pkt_len
8	tcp.flags	tcp.flags
9	frame.time_delta	frame.time_delta
10	tcp.len	tcp.len
11		ip.flags
12		tcp.flags.fin
13		ip.flags.df

wise in the case of both C4.5 and SBB based GP. The summary for AdaBoost is not as straight-forward and will therefore be limited to the total set of attributes utilized, independent of class.

Tables 6, 7 and 8 summarize these findings for the case of AdaBoost, SBB based GP and C4.5 respectively. SBB based GP clearly uses a lower total count of attributes relative to either AdaBoost or C4.5, and a lower count of attributes for SSH detection. Conversely, C4.5 uses the largest set of attributes as a whole or class-wise. Each classifier also identifies attributes unique to their solution. For example, SBB based GP is the only model to choose three attributes out of a possible four based on the TCP protocol – possibly because SSH applications run on TCP; while AdaBoost utilized the UDP attribute and C4.5 utilized the frame time attribute which is specific to a Network domain – potentially resulting in the poor DARPA data set generalization. Moreover, SBB is the only model to make use of '*ip.flags.df*' for SSH detection. This attribute may give more insight about the relation between encrypted traffic and keeping the integrity of packet (same size), so the decryption of the packet can be done correctly. Also of interest is the low level of overlap in shared attributes, with only 2 of 4 attributes shared between C4.5 and SSH under SSH detection and 5 of 11 attributes under non-SSH detection (w.r.t. SBB based GP attribute selection). What is certainly clear, however, is that a significant degree of preference exists in attribute selection relative to the machine learning model; thus, attempting to provide a limited 'hand crafted' set of attributes is likely to be counter-productive.

In terms of solution complexity, we note that AdaBoost generates 15 rules for SSH traffic and 16 rules for Non-SSH traffic; whereas C4.5 employs 527 rules for SSH classification and 693 rules to classify non-SSH traffic. Conversely, SBB based GP uses 2 individuals for SSH classification and 4 for non-SSH. We also note that instruction counts for the SSH classifying individuals was 4 and 8 instructions respectively; whereas individuals engaged in classifying non-SSH utilized 4, 12, and 11 (two off) instructions. In short, the simplicity of SBB solutions does not appear to be traded off for classifier complexity, in effect emphasizing the significance of support for problem decomposition in this application.

6. CONCLUSIONS

Previous research on traffic classification indicated that embedded paradigms such as C4.5 and AdaBoost provided better classification performance than methods without this capacity. In this work, we are able to take this concept further by introducing a model for learning problem decomposition in classification that explicitly associates independent subsets of exemplars with models identified during training. Such a team-based model of learning is not a weak learner, thus solutions take the form of a small number of simple programs with an explicitly non-overlapping behavioral trait and independent attribute subspaces. Such a methodology is able to provide solutions that are competitive under independent training and test data sets, while returning solutions that are potentially capable of higher throughputs of data than provided under a single 'monolithic' classifier per binary classification model (such as C4.5). Thus, solutions from C4.5 utilized deep decision trees, and are expensive to evaluate from a throughput perspective. Solutions located by AdaBoost were more robust than C4.5 under independent test conditions, but not as effective under the network from which training data was collected. Moreover, the weak learner methodology implicit in AdaBoost makes it much more difficult to associate attributes with class, reducing transparency of the resulting solution.

Future work will follow similar lines to perform more tests on different and/or larger data sets in order to continue to test the robustness of the classifiers as well as exploring the appropriateness of other machine learning algorithms. We are also interested in defining a framework for generating 'good' training data sets, where this might include combining training data from multiple independent sources. Evaluation under other encrypted applications as well as exploring the possibilities for integrating our approach with approaches employing host based behavior are also of interest.

Acknowledgments

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

7. REFERENCES

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

[2] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.

[3] R. Alshammari and A. Zincir-Heywood. A preliminary performance comparison of two feature sets for encrypted traffic classification. *Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems CISIS'08*, pages 203–210, 2009.

[4] R. Alshammari and A. N. Zincir-Heywood. Investigating two different approaches for encrypted traffic classification. In *PST '08: Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust*, pages 156–166, Washington, DC, USA, 2008. IEEE Computer Society.

[5] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic classification on the fly. *SIGCOMM Comput. Commun. Rev.*, 36(2):23–26, 2006.

[6] E. de Jong. A monotonic archive for pareto-coevolution. *Evolutionary Computation*, 15(1):61–93, 2007.

[7] J. Doucette and M. Heywood. Gp Classification under Imbalanced Data Sets: Active Sub-sampling and AUC Approximation. In *European Conference on Genetic Programming*, volume 4971 of *Lecture Notes in Computer Science*, pages 266–277, 2008.

[8] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286, New York, NY, USA, 2006. ACM Press.

[9] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. Acas: automated construction of application signatures. In *MineNet '05: Proceeding of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 197–202, New York, NY, USA, 2005. ACM Press.

[10] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 229–240, New York, NY, USA, 2005. ACM Press.

[11] l7 filter, last accessed March, 2008. http://l7-filter.sourceforge.net/.

[12] P. Lichodzijewski and M. I. Heywood. Coevolutionary bid-based Genetic Programming for problem decomposition in classification. *Genetic Programming and Evolvable Machines*, 9(4):331–365, 2008.

[13] P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with Symbiotic Bid-based Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 363–370, 2008.

[14] A. McIntyre and M. Heywood. Cooperative problem decomposition in Pareto competitive classifier models of coevolution. In *European Conference on Genetic Programming*, volume 4971 of *Lecture Notes in Computer Science*, pages 289–300, 2008.

[15] A. W. Moore and K. Papagiannaki. Toward the accurate identification of network applications. In *Passive and Active Network Measurement: Proceedings of the Passive &Active Measurement Workshop*, pages 41–54, 2005.

[16] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 50–60, New York, NY, USA, 2005. ACM Press.

[17] PacketShaper, last accessed March, 2008. http://www.packeteer.com/products/packetshaper/.

[18] C. D. Rosin and R. K. Belew. New methods for competitive coevolution. *Evolutionary Compuatation*, 5:1–29, 1997.

[19] W. Software. http://www.cs.waikato.ac.nz/ml/weka/.

[20] SSH. http://www.rfc-archive.org/getrfc.php?rfc=4251.

[21] G. M. Weiss and F. J. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *J. Artif. Intell. Res. (JAIR)*, 19:315–354, 2003.

[22] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *SIGCOMM Comput. Commun. Rev.*, 36(5):5–16, 2006.

[23] Wireshark, Last accessed Sep, 2008. http://www.wireshark.org/.

[24] C. Wright, F. Monrose, and G. M. Masson. Hmm profiles for network traffic classification. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 9–15, New York, NY, USA, 2004. ACM Press.

[25] C. V. Wright, F. Monrose, and G. M. Masson. On inferring application protocol behaviors in encrypted network traffic. *J. Mach. Learn. Res.*, 7:2745–2769, 2006.

[26] Y. Zhang and V. Paxson. Detecting back doors. In *In Proceedings of the 9th USENIX Security Symposium*, pages 157–170. ACM Press, 2000.