Can Encrypted Traffic be identified without Port Numbers, IP Addresses and Payload Inspection?

Riyad Alshammari, A. Nur Zincir-Heywood

Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada, B3H 1W5
{riyad,zincir}@cs.dal.ca

Abstract

Identifying encrypted application traffic represents an important issue for many network tasks including quality of service, firewall enforcement and security. Solutions should ideally be both simple – therefore efficient to deploy – and accurate. This paper presents a machine learning based approach employing simple Packet Header feature sets and statistical flow feature sets without using the IP addresses, source/destination ports and payload information to unveil encrypted application tunnels in network traffic. We demonstrate the effectiveness of our approach as a forensic analysis tool on two encrypted applications, Secure SHell (SSH) and Skype, using traces captured from entirely different networks. Results indicate that it is possible to identify encrypted traffic tunnels with high accuracy without inspecting payload, IP addresses and port numbers. Moreover, it is also possible to identify which services run in encrypted tunnels.

Key words:
Encrypted traffic identification, Packet, Flow, Security, Supervised learning, Efficiency, and Performance measures

1. Introduction

The accurate identification of network traffic according to the application type represents a challenging decision making activity that is not only important for network management tasks such as managing bandwidth budget and ensuring quality of service objectives for critical applications but also important for defense applications since it can facilitate the assessment of security threats. Such a system is particularly useful from a law enforcement application perspective since most of the time users with malicious intentions would try to hide their behavior either in encrypted or covert tunnels. Thus, systems that can classify encrypted traffic represent a first step in identifying such malicious users. In this case, establishing the classification of traffic types can actually reflect the current utilization of applications and services in a given traffic trace. This in return can help law-enforcement units to make a case to investigate 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. Traditionally, two approaches are used to identify IP network traffic: the first approach is to use 'well-known' Transmission Control Protocol (TCP) and/or User Datagram Protocol (UDP) port numbers (visible in TCP or UDP header) while the second approach includes more sophisticated techniques based on 'deep packet inspection within TCP or UDP payload (visible payload) looking for specific protocol signatures. Each approach relies on some assumptions that are no longer accurate and has many disadvantages. The first approach assumes most applications always use well-known port numbers registered by the internet assigned numbers authority (IANA) [1]. However, this assumption becomes increasingly inaccurate when applications use nonstandard ports to by-pass firewalls or circumvent operating systems restrictions. New applications such as Skype have not registered port numbers with IANA and assign port numbers dynamically. Moreover, the same port number can be used to transmit multiple applications, most notably port 80. Moore and Papagiannaki showed that classification based on IANA port list is correct 70% of the time [2]. Furthermore, Madhukar and Williamson confirmed that port number analysis misclassify 30-70% of their flow traffic [3].

On the other hand, the second approach assumes the access to the payload is always visible to inspect the payload of every packet. This technique can be extremely accurate when the payload is not encrypted. Sen et al. demonstrates that classifying Peer-to-Peer (P2P) traffic based on payload signatures could reduce false positive and false negative rates by 5% [4]. Moreover, Moore and Papagiannaki showed that using the entire payload can correctly classify 100% of packets [2]. However, the success of deep packet inspection is losing ground since encrypted applications such as Secure SHell (SSH) [5] or Skype Voice over Internet Protocol (VoIP) traffic [6] imply that the payload is opaque. Furthermore, governments may regulate the use of payload and enforce constrains on the use of payload since it can violate some of organizational privacy policies or go against related privacy legislation. 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 identification of efficient and effective classifiers. Different research groups have employed expert systems or various machine learning techniques such as Hidden Markov models, Naïve Bayesian models, AdaBoost, RIPPER, Decision Trees or Maximum Entropy methods to investigate this problem [7, 8, 9, 10, 11, 12, 2, 13, 14, 15]. Moreover, the limitations of port and payload based analysis have motivated the use of transport/flow layer statistics for traffic classification [16, 17, 18]. These techniques rely on the observation that different applications have distinct behavior patterns 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 mail, more work is necessary to distinguish encrypted applications accurately.

In this work, we have focused on the identification of encrypted traffic as a forensic analysis tool, where SSH and Skype are used as case studies. SSH is typically used to login to a remote computer but it also supports tunneling, file transfers and forwarding arbitrary transmission control protocol ports over a secure tunnel. On the other hand, Skype is a proprietary P2P VoIP application. 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, which can be used as a forensic analysis tool that distinguishes SSH traffic from non-SSH traffic and Skype from non-Skype traffic robustly without using IP addresses, TCP/UDP 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, too. To the best of our knowledge, none of the aforementioned works neither explored the robustness of the machine learning algorithms employed nor specifically focused on different encrypted traffic. Thus, in this paper, a machine learning based approach is employed to specifically identify SSH and Skype traffic. To this end, three machine learning algorithms, AdaBoost, C4.5 and GP, are evaluated to automatically generate signatures to identify Skype and SSH tunnels robustly.

Indeed, such an approach immediately raises several fundamental questions, including: how to establish the data on which 'general' – as opposed to network specific – classification signatures are identified, what representation and corresponding features to assume, and what representation the model of the classification should assume to satisfy both real-time (potentially) 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 (robustness). Issues of data representation are addressed by employing packet header based features and flow 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] but can still be achieved in near real-time.

The main research contributions of this paper are:

1. Exploring the limits of employing machine learning algorithms – C4.5, GP and AdaBoost – in order to classify encrypted traffic, specifically SSH and Skype, without using IP addresses, port numbers and payload data.
2. Demonstrating the ability to classify encrypted traffic using only one packet without any temporal information, i.e. packet header based features.
3. Exploring the performance of classifying encrypted traffic when temporal information is used, i.e. Flow based features.

[1] Flows are derived from a 5-tuple consisting of protocol (TCP/UDP), 'forward' and 'backward' IP addresses and corresponding port numbers. When IP numbers match within a finite temporal window 'flow' statistics are calculated.

4. Investigating the encrypted tunnel to identify the application types protected by the encrypted tunnel.
5. The analysis of which features are related to the classification target – encrypted tunnel.
6. The analysis of the robustness (generalization) of the signatures generated based on a series of tests performed on real network traffic that demonstrate the effectiveness of the proposed technique.

The rest of this paper is organized as follows. Summary of the two encrypted applications are discussed in Section 2. Related work is discussed in Section 3. Section 4 introduces the machine learning algorithms employed. Section 5 details the data sets, features and the evaluation method. Section 6 presents the experimental results. The analysis of the solutions is detailed in Section 7. Conclusions are drawn and future work is discussed in Section 8.

2. Overview of Encrypted Applications Deployed

In this paper, we have focused on the identification of two encrypted traffic applications, namely SSH and Skype, to validate our proposed approach.

2.1. Overview of SSH Application

SSH is an application that enables a user to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. SSH provides strong authentication and secure communications over unsecured tunnels. Moreover, it provides secure X11 connections and secure forwarding of arbitrary TCP connections [19, 20]. It is intended as a replacement for rlogin, rsh and rcp, where these commands are vulnerable to different kinds of attacks.

SSH consists of three layers. Each layer has a certain task to secure the connection between two hosts. These layers are as the following:

- **The connection layer** runs on top of both SSH transport and user authentication layers [21]. This layer can open many tunnels by multiplexing the secure established connection. Each tunnel is able to transfer data (packets) in both directions. This layer allows SSH to be used in different ways such as a terminal session, forwarding X11 information, transferring files, and creating tunnels.

- **The user authentication layer** runs on top of the transport layer [22]. This layer handles the client authentication by using several methods such as public key-based authentication (RSA or DSA) or password authentication.

- **The transport layer** secures the communication between two hosts during/after authentication [23]. It runs on top of the TCP/IP and handles the security through encryption/decryption of data (packets) and server authentication. Moreover, this layer is responsible for key exchange and encryption algorithms set up.

The SSH protocol allows the tunneling (port forwarding) of any TCP traffic stream on top of SSH. The SSH tunnel has the capability to provide confidentiality and integrity to any TCP traffic stream by encrypting and tunneling the stream from SSH client to SSH server to avoid network traffic being sent in clear-text or to avoid firewall restrictions. Therefore, users can tunnel unwanted/blocked traffic by the network administrators that potentially can expose their local network to viruses and worms. For example, users can tunnel any traffic by using this command:

$> ssh - g - L < localport >:< remotehost >:< remoteport > hostname$

This command sets up a secure connection from a local host to a remote host. Moreover, the port forwarding can be done on the other direction (remote forwarding). Consequently, network administrators that allow SSH would lose any control over what the users tunnel over SSH. In short, SSH can be used on any port (not necessarily port 22) to evade detection and bypass firewalls [24]. Thus, as discussed earlier, our aim is to investigate whether SSH can be detected without using port numbers, IP addresses and payload information.

2.2. *Overview of Skype Application*

Skype [6] is a very popular P2P VoIP application developed in 2002. Skype allows its users to communicate through voice calls, audio conferencing and text messages. Although Skype client provides similar functions as MSN and Yahoo instant message applications, the fundamental protocol and techniques it operates are completely different. Since Skype protocols are proprietary and an extensive use of cryptography are implemented by the Skype developers, understanding the Skype protocol is a difficult task. Moreover, Skype employs a number of methods to circumvent NAT and firewall restrictions [25], which increase the difficulty of understanding Skype protocol.

Skype is based on a P2P architecture except a user's authentication is performed based on a central architecture (client-server model via public key mechanisms). After authentication is completed, all communication is performed on the P2P network. Therefore, user information and search queries is stored and broadcasted in a decentralized approach. On the P2P network, there are two types of nodes, ordinary nodes (hosts) and supernodes. An ordinary node is a Skype client that can be used to make communication through the service provided by Skype. On the other hand, any node on the P2P network with sufficient CPU power, memory and network bandwidth is a suitable candidate for a supernode. A supernode is part of the decentralized Skype network that can ease the routing of Skype traffic to bypass NATs and firewalls. Moreover, ordinary hosts have to connect to a supernode and register with the Skype login server in order to join the P2P network.

Skype uses TCP or UDP at the transport layer to provide services to users such as voice and video calls, file transfer, chat and conference calls. For network communications, Skype mostly prefers UDP protocol. The communication among peers (users) on the P2P network is established via IP paradigm. However, Skype has the ability to route traffic via supernodes to circumvent the NATs and firewall restrictions. A more detailed description of Skype protocol and internals can be found in [25, 26].

3. Literature Review on Network Traffic Identification

Most of the existing research in the literature focuses on classification of well-known applications such as Web and mail. However, not much attention is paid to the classification of encrypted traffic. In the literature, Zhang and Paxson [27] present one of the earliest studies of techniques based on matching patterns in the packet payloads. Moore et al. [13, 2] used Bayesian analysis to classify flows into broad categories such as bulk transfer, P2P or interactive. Haffner et al. employed AdaBoost, Hidden Markov, Naïve Bayesian and Maximum Entropy models to classify network traffic into different applications [11]. Their results showed AdaBoost performed the best on their data sets; with an SSH detection rate of 86% and false positive rate of 0%, but they employed the first 64 bytes of the payload. Since the encryption of SSH data starts after the SSH handshake, analyzing the first 64 bytes of the payload (includes the not-encrypted part) provided them a good signature to classify SSH. Karagiannis et al. proposed an approach that does not use port numbers or payload information on well-known applications that are not encrypted [16]. However, their approach relies on information about the behavior of the hosts on the network. Thus, they cannot classify individual flows or connections. More recently, Wright et al. investigated the extent to which application protocols can be identified using only packet size, timing and direction information of a connection [15, 28]. They employed a k-nearest neighbor (kNN) and hidden markov model (HMM) learning systems to compare the performance. However, their performance on SSH classification is only 76% detection rate and 8% false positive rate. Bernaille and Teixeira employed first clustering and then classification to the first few packets in each connection to identify Secure Socket Layer (SSL) connections [29]. They use the first four packets of a TCP connection and represent it using 5-tuple (destination/source IP addresses, destination/source port numbers and protocol) and the packet size. However, they employed port numbers for classification which may cause problems when applications use port numbers dynamically. Montigny-Leboeuf 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 [12]. Based on these attributes, rules are formed manually to classify different application traffic. Results reported on SSH traffic classification showed 79% detection rate and 5% false positive rate (and 13% of flows were unrecognized) [12]. On the other hand, Li et al. employed C4.5 algorithm to classify network applications [30]. They use the first five packets of a connection. They reported a good performance for an online real time classification of network traffic. However, when they tested the trained model on a data set different than the training data set, the performance of their classifier dropped significantly (e.g. on the interactive applications, such as SSH, the Recall

was less than 30%). Palmieri et al. use nonlinear recurrence plot-based approach based on two flow features, which are average packet length and inter-arrival time variance, to classify traffic flows [31]. Their results on SSH showed ≈89% True Positive rate. Hu et al. employ a profile based approach with a two-level matching method, host-level matching and flow-level matching, to identify P2P flow traffic where they used BitTorrent and PPlive as two case studies. Their flow features are based on the five flow tuples {srcIP, destIP, srcPort, destPort, protocol} and statistical flow features. They used Apriori algorithm to achieve a compact set of flow patterns and build their rule sets using maximal association rules. They obtained an average accuracy for PPlive and BitTorrent when the validation data set is different than the training data set of ≈98% and ≈97% for PPlive over TCP and UDP respectively; and ≈94% and ≈96% for BitTorrent over TCP and UDP; respectively [32]. However, they do not report true positive and false positive rates.

To the best of our knowledge, the focus on the literature for detecting VoIP traffic is on Skype traffic because Skype is one of the most commonly used VoIP applications (Skype has 246 million users and around 10 million users are logged in online at any given time [33]). Therefore, Skype analysis has become popular in the last few years, in part due to the combination of the encrypted operation and dynamic nature of the port assignment making traditional methods of traffic identification redundant. Baset and Schulzrinne present an analysis of the Skype behavior such as login, network address translation (NAT) and firewall avoidance, and call setting up under three different network arrangements [25]. Suh et al. concentrate on the classification of relayed traffic and monitored Skype traffic as an application using relay nodes [34]. Relay node is part of the decentralized Skype network that can ease the routing of Skype traffic to bypass NATs and firewalls. They used several metrics based on features such as inter-arrival time, bytes size ratio and maximum cross correlation between two relayed bursts of packets to detect Skype relay traffic. Their results (a 96% true positive and 4% false positive) show the technique is reliable in recognizing relayed Skype sessions but it might not be appropriate to classify all Skype VoIP traffic. Bonfiglio et al. introduced two approaches to classify Skype traffic [35]. The first approach is to classify Skype client traffic based on Pearson's Chi-Square test using information revealed from the message content randomness (e.g. the FIN and ID fields) introduced by the cypher and the header format. Their second approach is to classify Skype VoIP traffic based on Naïve Bayesian Classifier using packet arrival rate and packet length. They obtained the best results when the first and second approaches were combined. They achieved approximately 1% false positive rate and between 2% to 29% false negative rate depending on the data sets.

In contrast to the related work, we specifically focus on encrypted tunnel identification without using the IP addresses, port numbers and payload data. Moreover, similar to the recent work to demonstrate the proposed system's capabilities, we have employed two case studies where SSH and Skype were chosen as the encrypted tunnels to identify. It should be noted here that in most of the previous work, the usage of port numbers, IP addresses and payload based features make the solutions less re-usable. In return, this causes a new feature set to be chosen for each application that needs to be identified. However, we aim to use a generic feature set and let the machine learning algorithms employed identify the subsets of it for classifying any given application. Recently, we have evaluated AdaBoost, Support Vector Machine, Naïve Bayesian, RIPPER and C4.5 using flow based features, where IP addresses, source/destination ports and payload information are not employed to classify encrypted traffic [8, 9, 36]. Results indicate that the C4.5 based approach outperforms other algorithms on the data sets employed. Furthermore, recently, we have compared Symbiotic Bid-based Genetic Programming (SBB-GP) based classifier against C4.5 [37] on SSH traffic classification. In that work, results show that GP based classifier was quite competitive with the C4.5 based classifier. Thus in this work, we aim to perform an investigation of C4.5, GP and AdaBoost based classifiers on the identification of SSH and Skype encrypted tunnels as well as explore their robustness in terms of evaluating them on different network traces from different institutions. Last but not the least, none of the previous work employing machine learning techniques to detect different application traffic investigated the robustness/generalization of such techniques to different encrypted applications and different network traces. What we mean by robustness here is that the signatures to classify/identify the encrypted traffic are generated on network traffic from one network but evaluated (tested) on network traffic, which are from completely different networks.

4. Learning Models

In this work, we are interested in the application of supervised machine learning (ML) based techniques to network traffic classification, specifically classification of encrypted traffic. The reason we took a ML based approach is

twofolds: (i) the need for automating the process of identifying such traffic in terms of automatically creating the signatures (rules) that are necessary to classify encrypted traffic as well as (ii) automating the process of selecting the most appropriate attributes for those signatures. The ML techniques require a number of steps. First, a matrix of instances versus features is needed to describe the data set. A vector of features describes each instance or record in a given trace/traffic file. The features are used as values to quantify different characteristic of the instance (network traffic) such as packet size or inter-arrival time. Second, a label is provided for each instance, which is the class description (network application type). Finally, ML needs to be trained using a data set (called training) and gives an output, which consists of the rules or the model it generates. This output can then be verified on a test data set (unseen instances). A more detailed explanation of ML and traffic classification can be found in [38].

In this section, we summarize the three machine learning techniques employed in this work. These are C4.5, AdaBoost and GP. The reason we have employed these machine learning algorithms are the following. Previous works have reported good performance of these learning models in their respective works [10, 11, 14]. In our own previous work we have observed these models to give the best solutions under different network conditions [7, 8, 36, 37, 39, 40]. Moreover, all of these learning models are capable of choosing the best attributes from a given set. This is an important property, given that we are interested in analyzing which attributes are the best from a set of all possible attributes. Last but not the least, all these learning algorithms can automatically generate solutions in the form of rules that are easy to understand by human experts. Again, an important property, given that system administrators or network engineers should be able to understand the signatures/rules generated in order to accept to employ them.

4.1. 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. 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. One possible function to measure impurity is entropy, equation 1 [41].

$$J_m = -\sum_{j=1}^{n} p_m^i \log_2 p_m^i \tag{1}$$

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 done. Indeed, this is locally optimal, hence has no guarantee on finding the smallest decision tree. In this case, the total impurity after the split can be measured by equation 2 [41]. In other words, when a tree is constructed, at each step the split that results in the largest decrease in impurity is chosen. This is the difference between the impurity of data reaching node m, equation 1, and the total entropy of data reaching its branches after the split, equation 2. A more detailed explanation of the algorithm can be found in [41].

$$J_m^{'} = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{k} p_{mj}^i \log p_{mj}^i \tag{2}$$

4.2. AdaBoost

AdaBoost, an acronym for Adaptive Boosting, was developed by Yoav Freund and Robert Schapire [42]. It is a meta-learning algorithm, which means that a strong classifier is built from a linear combination of weak (simple) classifiers [2]. 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

[2]Weak or simple implies that classification performance (on a balanced data set) might only be slightly better than 50%.

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 runs for a given number of iterations, T. At each round, there are two possible outcomes: (i) A new decision stump is added to the structure with a calculated weight or vote that reflects its influence on the overall classification process. (ii) The distribution of the training instances is modified to give higher importance to the incorrectly classified data so that the next weak classifier is forced to eliminate/minimize the training errors in the next iteration in order to improve the overall performance.

Specifically, classifiers, h, are iteratively added to the set of weak learners at each training epoch, t. Thus, after selecting an optimal classifier h_t for the distribution D_t, the training examples x_i that the classifier h_t identified correctly are weighted less and those that it identified incorrectly are weighted more. Therefore, when the algorithm is testing the correct set of weak classifiers on the distribution D_{t+1}, it will construct a new weak classifier that better identifies those examples that the previous set of weak classifiers missed. Once this process has been completed the resulting structure returned by AdaBoost is the final (strong) classifier with a weighted majority vote of T weak classifiers. This is defined to be the sequence of decision stumps that can best classify the training set with the given features. In short, 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 description of AdaBoost learning model can be found in [41].

4.3. Team-based Genetic Programming

In this work, the form of genetic programming employed is based on the Symbiotic Bid-Based (SBB) paradigm of team based GP. The SBB framework makes extensive use of coevolution [43], with a total of three populations involved: a population of points, a population of learners, and a population of teams (Fig. 1). 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 case of a population wide competition between bids. 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 [44]. 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 [45], whereas the concept of Pareto competitive coevolution is used to retain points of most relevance to the competitive coevolution of teams.

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. 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 [44]. 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).

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 [46]. An analogous process is repeated for the case of team selection, with $(M_{size} - M_{gap})$ icndividuals copied into the next generation. 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 [43].

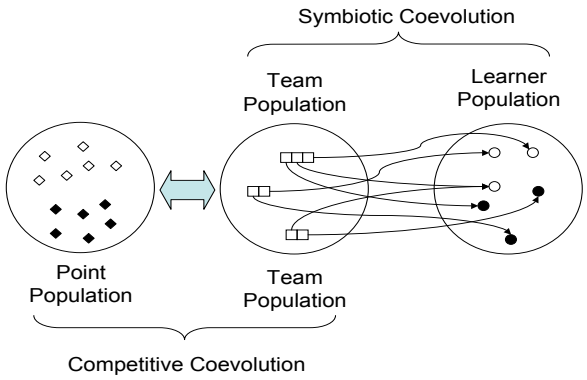


Figure 1: Architecture of Symbolic Bid-based GP: Point to team populations are competitive, Team to learner populations are symbiotic (cooperative).

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 [43] and consists of eight opcodes ($\{cos, exp, log, +, \times, --, \div, \%\}$) operating on up to eight registers, as per a linear GP representation. A more detailed description of SBB based GP learning model can be found in [43].

5. Evaluation Methodology

As discussed earlier, C4.5, AdaBoost and SBB-GP learning models will be employed to automate the process of choosing attributes/features and forming data driven signatures under the packet based and flow based encrypted traffic classification problem. In order to evaluate the performance of the proposed system, two case studies, namely, the identification of SSH encrypted tunnels and the identification of Skype encrypted tunnels are investigated. Moreover, solution robustness is assessed by training on a data set from one location (hereafter denoted University) but by testing on data sets from other locations (hereafter AMP, MAWI and DARPA99, which were captured in on 2005, 2006, and 1999, respectively). On the other hand, solution robustness for Skype is assessed by training on the data set denoted as University but tested on a data set denoted as Italy. Please note that we could not use the same testing traces for both SSH and Skype tunnel identification because there was no Skype traffic in the AMP, MAWI and DARPA99 traces. The properties of the data sources are as follows:

- **University data sets** 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 largest universities in the Atlantic region of Canada. There are more than 15,000 students and 3,300 faculty and staff. The UCIS is responsible for all the networking on the campus, which includes more than 250 servers and 5000 computers. Dalhousie traces were captured for 8 h from the main Internet link. Given the privacy related issues, 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.

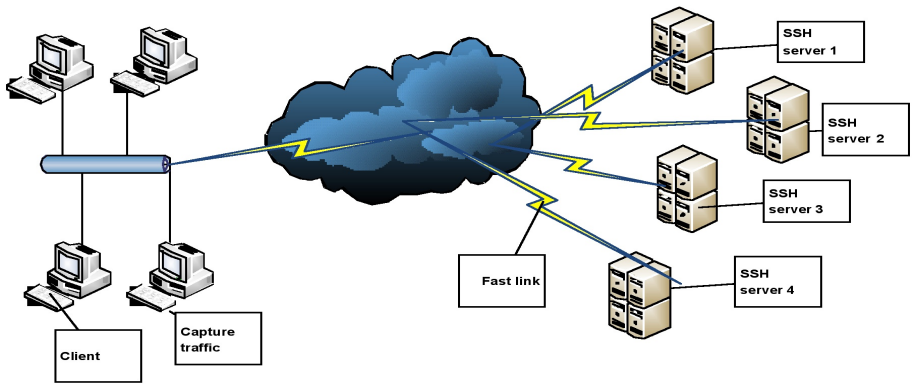


Figure 2: The NIMS testbed network used for traffic generation and capturing

- **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) [47] and MAWI (Measurement and Analysis on the WIDE Internet) [48] data repositories. In the case of NLANR repository, we used AMP*traces (AMP312, AMP313, AMP314, AMP316, AMP317, AMP318, AMP320, AMP321) since there is a reasonable amount of SSH traffic in them. AMP data sets are stratified random samples captured in 2005 at NAP-of-the-Americas in Miami (1000Mbps link). In the case of MAWI repository, we used traces, which also consisted of reasonable amount of SSH traffic (04121400, 05141400). MAWI traces are daily traces captured from a trans-Pacific line (18Mbps CAR on 100Mbps link) in 2006.

- **NIMS data sets** consist of packets collected internally at the authors' research testbed network. In this case, different network scenarios are emulated using one or more computers to capture the resulting traffic. SSH connections are generated by connecting a client computer to four SSH servers outside the testbed via the Internet, Fig. 2. The following six SSH services: (i) Shell login; (ii) X11; (iii) Local tunneling; (iv) Remote tunneling; (v) SCP; and (vi) SFTP are also run and the traffic generated is captured. Application behaviors (background traffic) such as DNS, HTTP, FTP, P2P (limewire), and telnet are also emulated on the testbed network.

- **DARPA99** data set consists of five weeks of traces generated at the MIT Lincoln Labs for Intrusion Detection Evaluation [49]. For each week, there are five network trace files (including the payload) that represent 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 identify encrypted tunnels such as SSH in given network traffic but not if we can detect intrusions. In this case, only the 'inside' sniffing data was employed.

- **Italy** data set consists of 96 h of Skype Traffic over TCP and UDP protocols [50]. The data set is captured on the main link at the Politecnico di Torino University campus. They classified Skype traffic based on deep packet inspection and per-host analysis using TCP Statistic and Analysis Tool (Tstat) and the traffic classification method described in [35] (as described in section 3). Furthermore, their second approach results in a classifier, which depends on Pearson's chi-square test using information from the message content (payload) and their third approach results in a classifier, which depends on Naïve Bayesian classification technique using packet arrival rate and packet length to classify the Skype traffic. The third approach is closer to our method however we have shown that C4.5 works much better than Naïve Bayesian in [36]. In this work, we employed their two Skype End-to-End call traces. The first trace, which is captured over UDP, consists of voice only calls as well as voice plus video calls. The second trace is captured over TCP and consists of SkypeOut traffic. Brief statistics on the traffic data collected are given in Table 1.

It should be noted here that the University traces are labeled by a commercial classification tool (PacketShaper), which is a deep packet analyzer [51], by the university's UCIS team. PacketShaper uses Layer 7 filters (L7) to classify

Table 1: Summary of traces used in this work

	University	**DARPA99**	**AMP**	**MAWI**	**NIMS**	**Italy**
Total Packets	337,041,778	16,723,835	332,064,652	76,543,335	34,808,433	41,985,540
MBytes	213,562	3,638	188,435	28,718	35,640	8,147
% of TCP packets	86.51%	88.6%	55.36%	85.37%	96.01%	5.62%
% of TCP bytes	91.03%	93.29%	72.05%	70.3%	98.77%	3.75%
% of UDP packets	13.33%	11.33%	33.6%	11.65%	3.76%	94.38%
% of UDP bytes	8.95%	6.43%	11.2%	5.5%	1.22%	96.25%
% of Other packets	0.16%	0.07%	11.04%	2.98%	0.23%	0.0%
% of Other bytes	0.02%	0.28%	16.75%	24.2%	0.01%	0.0%
Total SSH Flows	19,384	72,094	427,448	19,016	14,681	N/A
Total non-SSH/Skype Flows	35,546,177	28,489,208	20,669,977	19,954,825	699,170	N/A
Total Skype Flows	8,664,137	N/A	N/A	N/A	N/A	389070

applications [52]. On the other hand, a port-based classifier is employed for labeling the AMP, MAWI and DARPA99 traces. This assumes IANA assignments and follows the approach taken in the previous works [17, 18, 11, 2, 14, 28], where such public traces without any ground truth are used. Given that AMP and MAWI data sets do not have payload, this is the only way we can label them. For these traces, we will assume that port based classification reflects the ground truth (even though we know that it does not). However, DARPA99 traces also have payload, so in this case, we were able to verify the labels for SSH traffic based on the non-encrypted handshake part of SSH in the payload. This enabled us to establish the ground-truth for DARPA99 traces. Finally, establishing the ground truth for NIMS traces was not a problem since we knew exactly, which applications were running in every experiment. Moreover, because we generated the NIMS traces on our testbed network, classifying applications running over SSH was also possible. We were able to label the different applications/services running in SSH tunnels. In this case, there are six different application labels of such: i) SCP – indicating a secure copy session; ii) SFTP – indicating a secure file transfer session; iii) LocalT – indicating a tunneling session to a local machine; iv) RemoteT – indicating tunneling session to a remote machine; v) X11 – indicating an X11 session; and vi) Shell – indicating an SSH remote login session (interactive terminal).

In this work, for SSH tunnel identification, we have used a subset of University traces as training and the rest of the University traces, public traces (AMP and MAWI) and DARPA99 traces (weeks 1 and 3) as testing. We performed subset sampling to limit the memory and CPU time required for training. Thus, we sampled the training data set from the Dalhousie traces since this represents the newest trace among the data sets employed. For SSH tunnel identification based on flow attributes, 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 12,246 flows, 6123 SSH and 6123 non-SSH. For SSH tunnel identification based on the packet header attributes, the training data set included the same packets that generated the flows. In total, the sample consists of 311,130 packets, of which 157,273 are SSH and 153,857 are non-SSH.

On the other hand, for Skype identification, Skype Training Sample is generated by sampling randomly selected (uniform probability) flows from different classes (FTP, SSH, MAIL, DNS, HTTP, HTTPS and Random UDP). Since there are not much TCP flows from University traces, we sample 13,306 TCP flows from Italy data sets and aggregated with the Skype Training Sample. In total, Skype Training Sample consists of 828,964 flows (472,784 Skype flows versus 356,180 non-Skype flows). For Skype tunnel identification based on the packet header attributes, Skype Training Sample is generated by sampling randomly selected (uniform probability) packets from the same classes used to sample the flow training data set. In total, Skype Packet header Sample consists of 600,000 balanced packets (Skype packets versus non-Skype packets). [3]

As discussed earlier, our approach to identifying encrypted traffic such as SSH and Skype is data-driven, we present all the possible attributes/features to the learning algorithms employed. In doing so, we aim to examine: (i)

[3]The data sets can be download in ARFF format based on the flow based features at http://www.cs.dal.ca/~riyad/

which features will be considered the most appropriate by each learning algorithm, and also, (ii) which features will be chosen by all of the learning models employed in this work. We believe that this subset of features will give us the most robust/generalized as well as the most appropriate ones that can be used on real life network traffic traces.

5.1. Feature selection – packet header based features

In this work, each packet is described in terms of 39 features, Table 2, where the underlying principle is 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 [53] is employed to process data sets and to generate features. As discussed earlier, we did not use the IP addresses, port numbers and payload data.

5.2. Feature selection – flow based features

For the flow based feature set, a feature is a descriptive statistic that can be calculated from one or more packets for each flow. To this end, NetMate [54] is employed to generate flows and compute 22 features, Table 3. Flows are bidirectional and the first packet seen by the tool determines the forward direction. We consider only UDP and TCP flows. Moreover, UDP flows are terminated by a flow timeout, whereas TCP flows are terminated upon proper connection teardown or by a flow timeout, whichever occurs first. The flow timeout value employed in this work is 600 seconds as recommended by the IETF [55].

5.3. An Open Source Tool to classify traffic – Wireshark

Wireshark [53], which was formerly known as Ethereal, is the most popular open-source, cross-platform network analysis tool. Network Packets can be analyzed by Wireshark either online or offline. Wireshark makes use of libpcap [56] library for packet sniffing. Further, Wireshark is available for different platforms such as Unix-based, Windows-based and Macintosh machines.

5.3.1. Evading Wireshark

The motivation of these experiments is to show the effectiveness of Wireshark as a network analysis tool. Since Wireshark can be used as a network analysis tool to label traffic according to the application types, the following experiments can show the ability of Wireshark to classify an encrypted application such as SSH. We ran the Wireshark system on the DARPA99 trace since it is the only trace with the payload (Dalhousie traces has been anonymized) to understand the true detection rate and false positive rate for the Wireshark based system. The motivation of this experiment is to show the effectiveness of Wireshark type of traffic analyzers, which inspect all the packet header fields (including the IP addresses and TCP/UDP port numbers) as well as the payload information, as traffic identification systems. Therefore, to fully demonstrate how the Wireshark uses signatures based on port numbers to label SSH traffic, we ran two experiments where we modified the port number in the SSH trace from port 22 to port 2200 in the first experiment and from port 22 to port 80 in the second experiment using tcprewrite (Unix tool to rewrite packets in pcap file), and then run Wireshark again. Fig. 3a shows the result of the first experiment, where the Wireshark tool failed to detect any of the SSH packets when the port number for SSH is changed from port 22 o port 2200. Fig. 3b shows the result of the second experiment, where the Wireshark tool classified SSH packets as HTTP packets, when the port number is changed from port 22 to port 80.

Furthermore, we used the Italy traces to see if the Wireshark system can detect Skype traffic running either on TCP protocol or UDP protocl. The results did not surprise us. Fig. 4 shows that Wireshark failed to detect any of the Skype traffic.

The above experiments illustrate that Wireshark depends on well-known port numbers to classify the traffic. Thus, a new approach is necessary that does not depend on port numbers on which the classifier/analyzer cannot be evaded just by changing the port number where the application runs.

Table 2: Packet Header based features employed, * Normalized by log

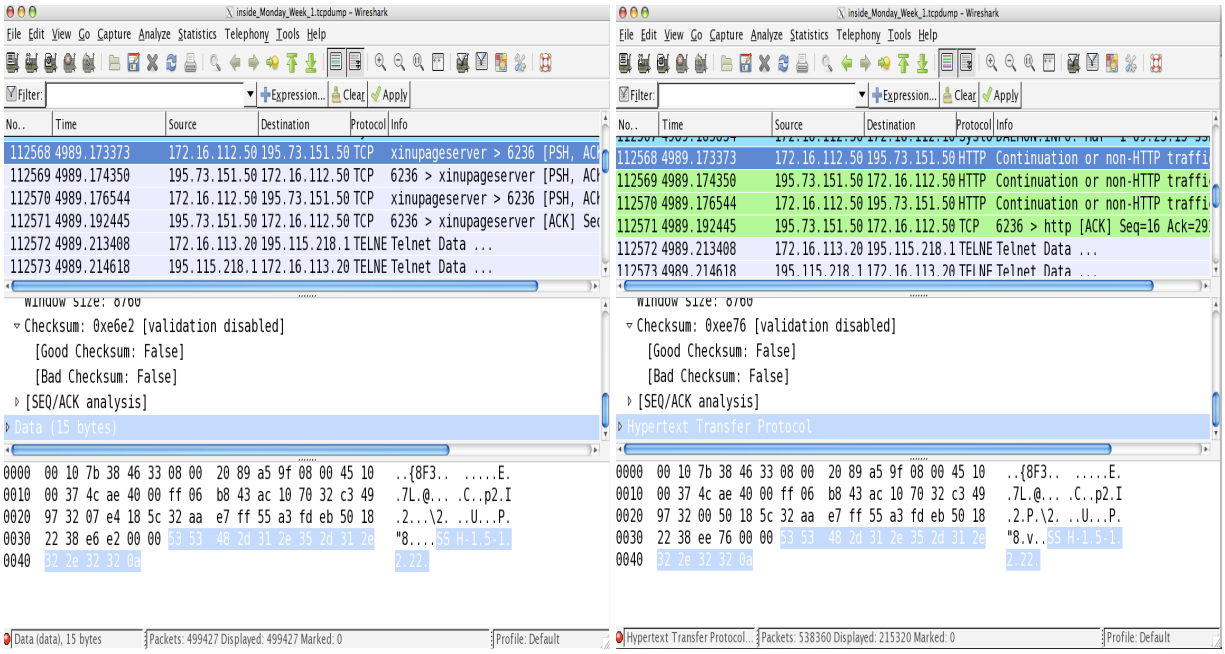
No.	Feature Name	Description
1	frame.time_delta	Delta time from previous captured packet
2	frame.pkt_len*	Packet Length
3	frame.len*	Frame Length
4	frame.cap_len*	Capture Length
5	frame.marked	Frame is marked
6	ip.len*	IP Header length
7	ip.flags	IP Flags
8	ip.flags.rb	IP Flags: Reserved bit
9	ip.flags.df	IP Flags: don't fragment
10	ip.flags.mf	IP Flags: More fragments
11	ip.frag.offset	IP Fragment offset
12	ip.ttl*	IP Time to live
13	ip.proto	IP Protocol
14	ip.checksum	IP Header checksum
15	ip.checksum_good	IP Header checksum is set to true (1)
16	ip.checksum_bad	IP Header checksum is set to false (0)
17	tcp.len*	TCP Segment Length
18	tcp.seq*	TCP Sequence number
19	tcp.nxtseq*	TCP Next sequence number
20	tcp.ack*	TCP Acknowledgement number
21	tcp.hdr_len*	TCP Header length
22	tcp.flags*	TCP Flags
23	tcp.flags.cwr	TCP Flags: Congestion Window Reduced
24	tcp.flags.ecn	TCP Flags: ECN-Echo
25	tcp.flags.urg	TCP Flags: Urgent
26	tcp.flags.ack	TCP Flags: Acknowledgment
27	tcp.flags.push	TCP Flags: Push
28	tcp.flags.reset	TCP Flags: Reset
29	tcp.flags.syn	TCP Flags: Syn
30	tcp.flags.fin	TCP Flags: Fin
31	tcp.window_size*	TCP Window size
32	tcp.checksum	TCP Checksum
33	tcp.checksum_good	TCP Checksum is set to true (1)
34	tcp.checksum_bad	TCP checksum is set to false (0)
35	udp.length*	UDP Length
36	udp.checksum_coverage*	UDP Checksum coverage
37	udp.checksum	UDP Checksum
38	udp.checksum_good	UDP Checksum is set to true (1)
39	udp.checksum_bad	UDP Checksum is set to false (0)

6. 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 (FPR). In this case, DR will reflect the number of SSH/Skype packets/flows correctly classified and is calculated using $DR = \frac{TP}{TP+FN}$; whereas FPR will reflect the number of non-SSH/non-Skype packets/flows incorrectly classified as SSH/Skype and is calculated using $FPR = \frac{FP}{FP+TN}$. Naturally, a high DR rate and a low FPR are the most desirable outcomes. False negative, FN, implies that SSH/Skype traffic is classified

Table 3: Flow based features employed

Protocol (proto)	Duration of the flow (Duration)
# Packets in forward direction (fpackets)	# Bytes in forward direction (fbytes)
# Packets in backward direction (bpackts)	# Bytes in backward direction (bbytes)
Min forward inter-arrival time (min_fiat)	Min backward inter-arrival time (min_biat)
Std deviation of forward inter-arrival times (std_fiat)	Std deviation of backward inter-arrival times (std_biat)
Mean forward inter-arrival time (mean_fiat)	Mean backward inter-arrival time (mean_biat)
Max forward inter-arrival time (max_fiat)	Max backward inter-arrival time (max_biat)
Min forward packet length (min_fpkt)	Min backward packet length (min_bpkt)
Max forward packet length (max_fpkt)	Max backward packet length (max_bpkt)
Std deviation of forward packet length (std_fpkt)	Std deviation of backward packet length (std_bpkt)
Mean backward packet length (mean_fpkt)	Mean forward packet length (mean_bpkt)



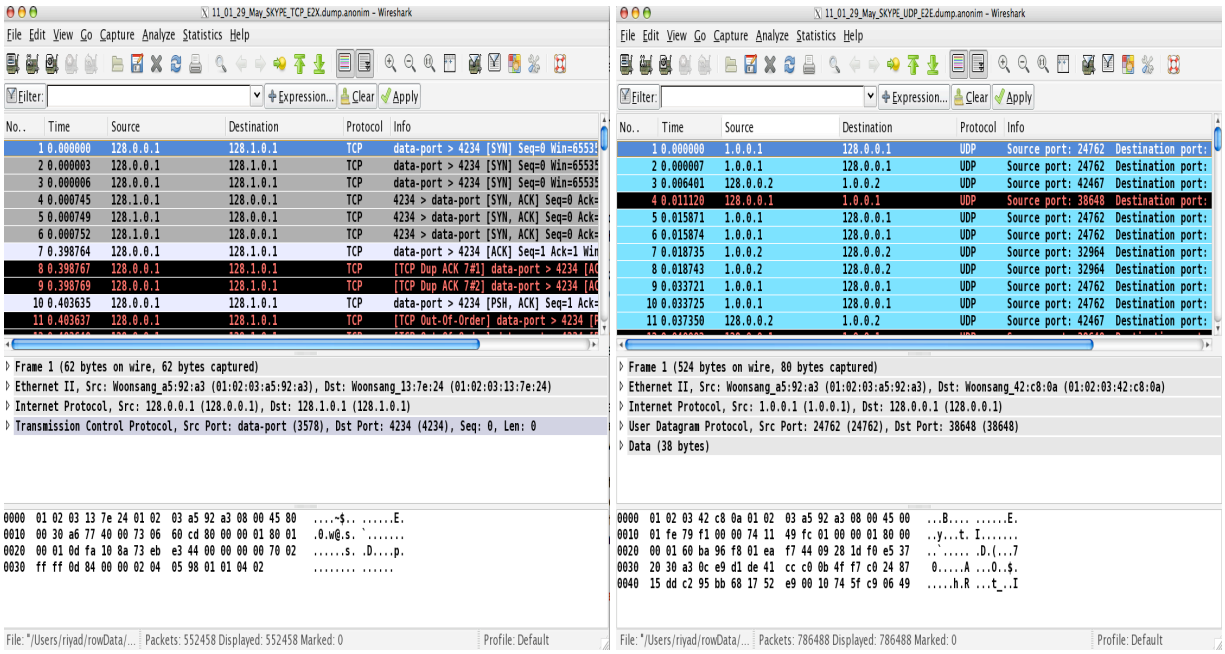
(a) Changed from port 22 to port 2020

(b) Changed from port 22 to port 80

Figure 3: Wireshark failed to classify SSH packets after modifying ports on the DARPA99 traces.

as non-SSH/non-Skype traffic, FP, false positive, implies that non-SSH/non-Skype traffic is classified as SSH/Skype traffic.

All three candidate classifiers are trained on the training data using fifty runs to generate 50 different models for each run. Weka [57] is employed with default parameters to run C4.5 and AdaBoost. Fifty runs of the C4.5 algorithm are performed using different confidence factors to generate different models for C4.5 and fifty runs of the AdaBoost algorithm are performed using different weight thresholds to generate different models for AdaBoost. The SBB-GP classifier default parameters are summarized in Table 4. Fifty runs of the SBB-GP algorithm are performed using different population initializations to generate different models. Such an approach ensures that our results are based on statistically significant experiments as opposed to one-off trials.



(a) Skype on TCP

(b) Skype on UDP

Figure 4: Wireshark failed to classify Skype packets on the Italy traces.

Table 4: SBB-GP model 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

6.1. Results of packet header experiments for SSH and Skype identification

In these set of experiments, the objective is to identify SSH and Skype on a packet per packet basis using only the features given in Table 2.

Fig. 5 summarizes solutions for SSH detection for the three machine learning algorithms on the (University) training traces and test traces. That is to say, all model construction takes place on the University Training partition. Testing evaluation is conducted under University Test and AMP, MAWI and DARPA99 traces where none were encountered during training. Naturally, the University Test partition will more closely reflect the training behavior than the AMP, MAWI and DARPA99 network traces.

Results show that the C4.5 based classification approach is much better than AdaBoost and SBB-GP algorithms employed in identifying the SSH traffic based on the training data set, Fig. 5. Moreover, in the case of C4.5, much lower variance (Table 5) implies that the corresponding solutions generalize to the wider case. On the other hand, results on the test traces show that on average AdaBoost is much better than other machine learning algorithms on the test data sets in terms of high DR and low FPR.

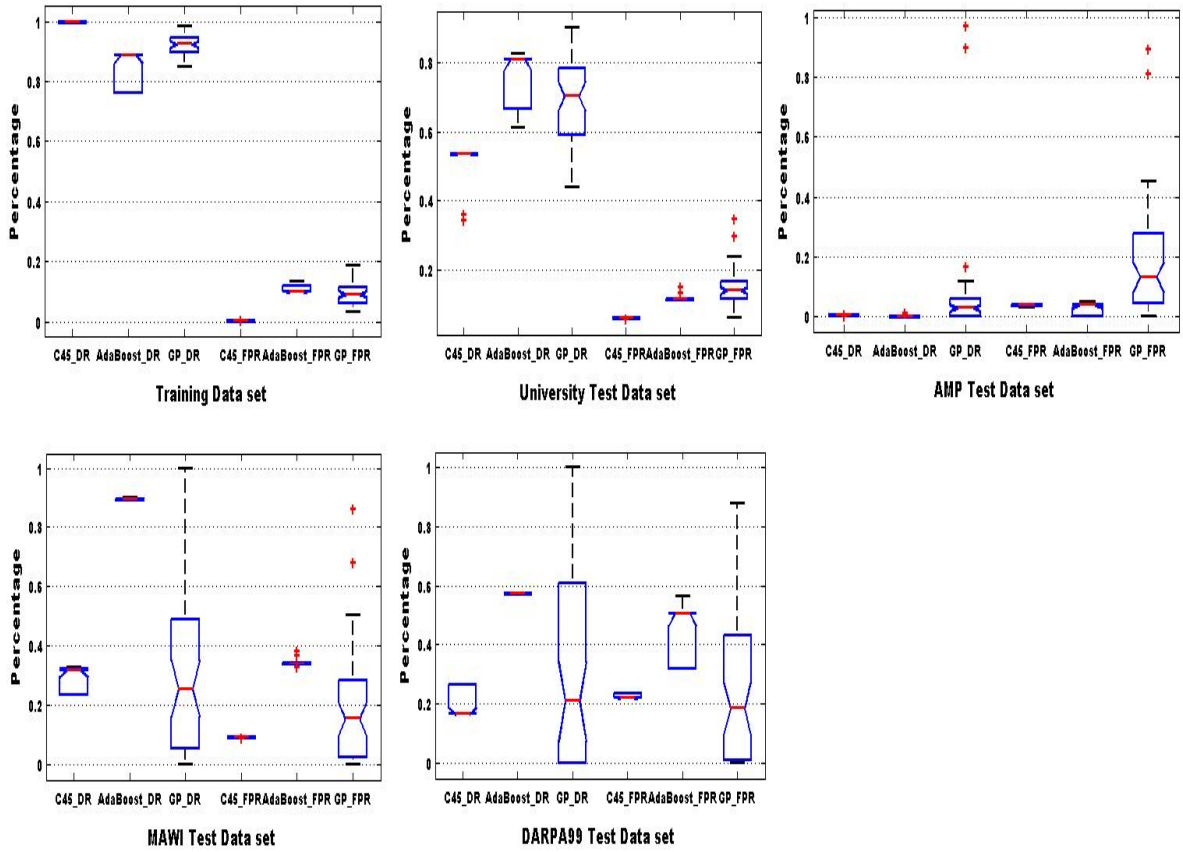


Figure 5: Results on Data sets for Packet Header based Feature set for SSH detection.

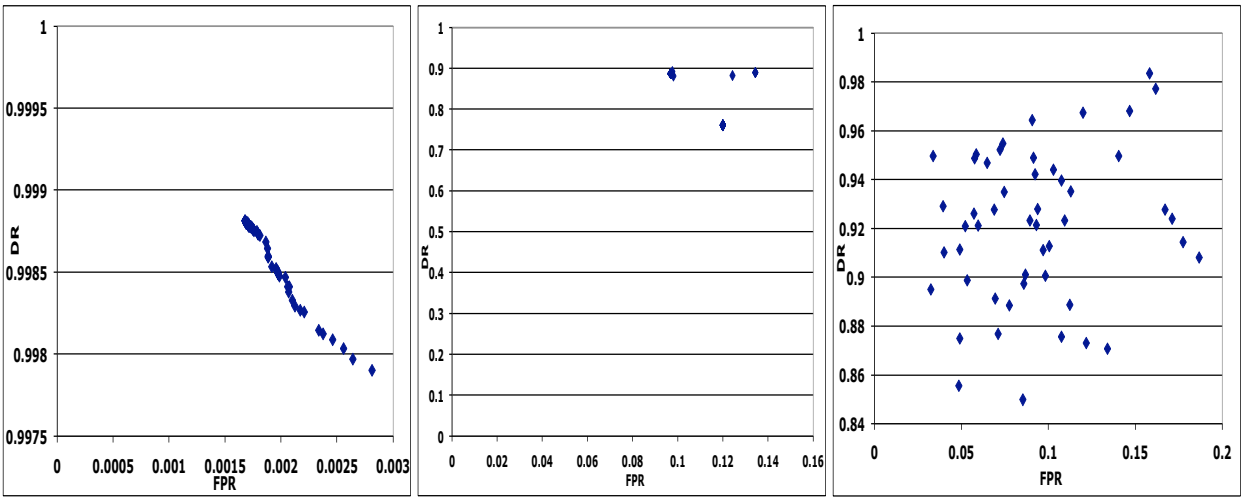
Table 5: Standard Deviation of Results on the training data for SSH detection based on packet header feature set.

	C4.5		AdaBoost		GP	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample (subset of university) x 50						
non-SSH	0.0003	0.0003	0.012	0.056	0.045	0.033
SSH	0.0003	0.0003	0.056	0.012	0.033	0.045

The training performances are plotted in ROC curves to decide on the best solution from each machine learning algorithm, Fig. 6 summarizes these results. There are nine that are non-dominated for GP, two that are non-dominated for AdaBoost and one that is non-dominated for C4.5. Then, the highest performance non-dominated solution based on DR and FPR on the training result for GP, AdaBoost and C4.5 are selected and then evaluated on the test data sets.

Results show that on the training data set, C4.5 appears to provide the stronger performance with consistently better FPR and DR. Introducing the entirely independent test sets – AMP, MAWI and DARPA99 – indicated that C4.5 had over-learnt the properties implicit in the training partition. Moreover, SBB-GP and AdaBoost were observed to provide best case performances under the independent test partitions DARPA99, MAWI and University Test data sets. All learning algorithms performed poorly under the AMP data sets.

Results show that the AdaBoost classifier performs better than the other classifiers on the majority of the data sets.



(a) C4.5 (b) AdaBoost (c) GP

Figure 6: ROC Curve plot for the three classifiers for training performance using Packet Header based Feature set for SSH detection (DR versus FPR)

Table 6: Best solution out of 50 runs for each Classifier on training and testing data sets for SSH detection based on packet header feature set.

	C4.5		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample						
non-SSH	0.998	0.0012	0.902	0.108	0.968	0.105
SSH	0.999	0.0017	0.892	0.098	0.895	0.032
University Traces						
non-SSH	0.941	0.464	0.885	0.19	0.943	0.295
SSH	0.536	0.0594	0.81	0.115	0.705	0.057
DARPA99 Traces						
non-SSH	0.777	0.835	0.495	0.429	0.644	0.041
SSH	0.165	0.223	0.571	0.505	0.959	0.356
AMP Traces						
non-SSH	0.959	0.995	0.96	0.996	0.828	0.939
SSH	0.005	0.041	0.004	0.04	0.061	0.172
MAWI Traces						
non-SSH	0.906	0.675	0.657	0.104	0.787	0.593
SSH	0.325	0.094	0.896	0.343	0.407	0.214

AdaBoost classifier achieves ≈81% DR and ≈12% FPR on Dalhousie traces and ≈90% DR and ≈34% FPR on the MAWI Test traces whereas SBB-GP achieves best result on Dalhousie traces with ≈83% DR and ≈16% FPR, 65% DR and ≈20% FPR on MAWI traces, and 47% DR and ≈3% FPR on the DARPA99 traces, Table 6. Moreover, the SBB-GP classifier was the most consistent performer across all test and training conditions, while also being competitive with AdaBoost under University traces (particularly in terms of SSH False Positive rate and SSH Detection rate). However, robustness results also show that FPRs of each learning algorithm are too high to employ SSH tunnel identification based on a single packet's attributes.

On the other hand, for Skype detection based on the packet header features, Fig. 7 lists the results for the three

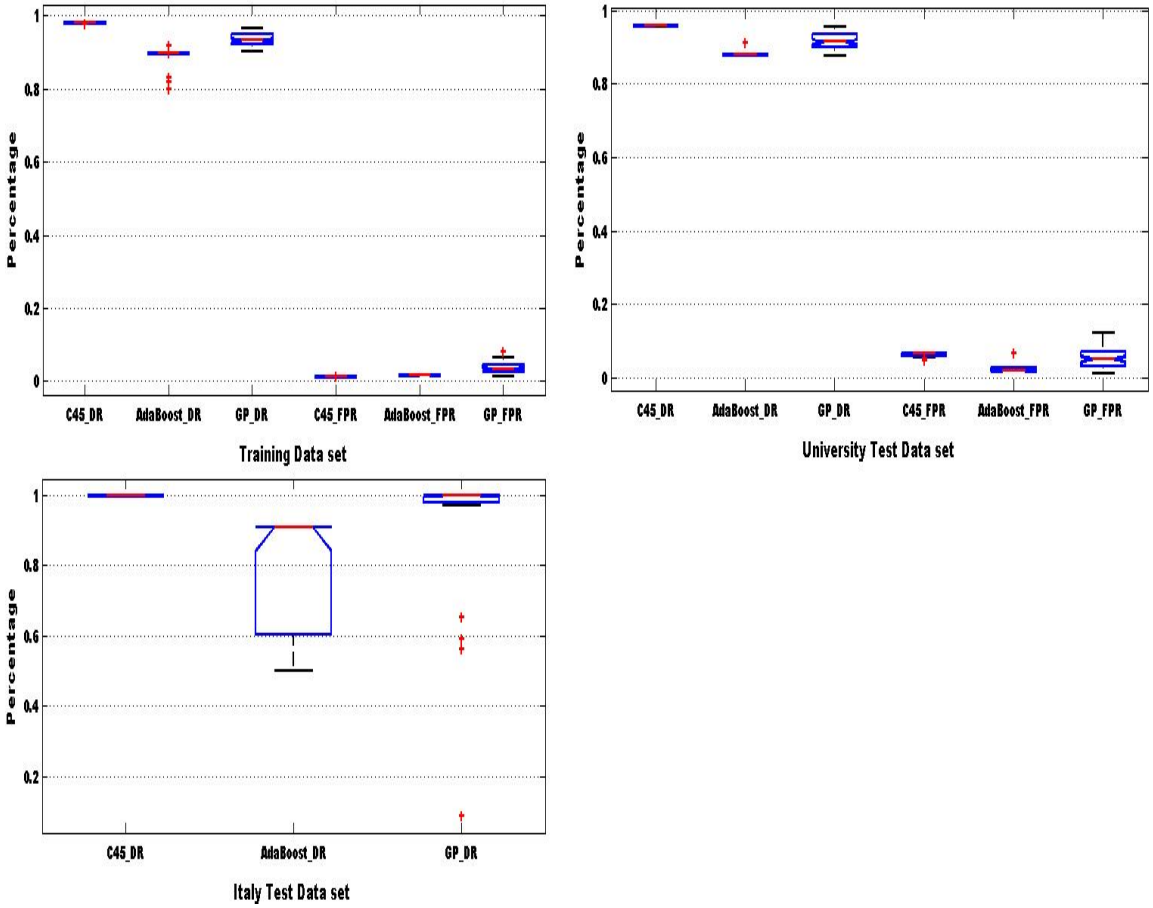


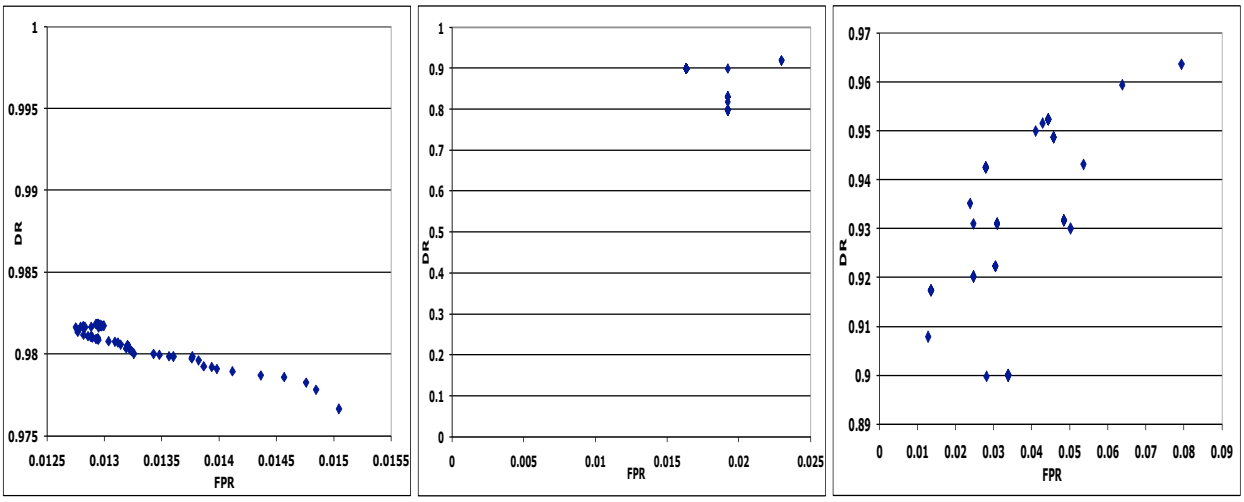
Figure 7: Results on Data sets for Packet Header based Feature set for Skype detection.

Table 7: Standard Deviation of Results on the training data for Skype detection based on packet header feature set.

	C4.5		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample (subset of university) x 50						
non-Skype	0.0004	0.0008	0.224	0.042	0.03	0.02
Skype	0.0008	0.0004	0.04	0.224	0.021	0.03

machine learning algorithms on the (University) training, and independent test traces. All models return high detection rate for Skype traffic on the training data set but C4.5 has much lower variance (Table 7) implies that the corresponding solutions generalize to the wider case, implicit in the test results. Moreover, C4.5 and SBB-GP appear to provide the strongest performance with consistently better FPR and DR on average on the test data sets.

In this case, the training performances are plotted in ROC curves to determine the best trained model for each machine learning algorithm. Fig. 8 summarizes these findings. For Skype, there are nine that are non-dominated for GP, four that are non-dominated for AdaBoost and four that are non-dominated for C4.5. We considered the highest performance solution in terms of high DR and low FPR out of these non-dominated solutions for GP, AdaBoost and C4.5 are then evaluated on the test data sets. Additionally, Table 1 illustrates that these traces indeed belong to significantly different networks. Therefore, we believe that only well generalized models are able to classify Skype



(a) C4.5 (b) AdaBoost (c) GP

Figure 8: ROC Curve plot for the three classifiers for training performance using Packet Header based Feature set for Skype detection (DR versus FPR)

Table 8: Best results out of 50 runs for each Classifier on training and testing data sets for Skype detection based on packet header feature set.

	C4.5		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample						
non-Skype	0.987	0.018	0.984	0.101	0.921	0.036
Skype	0.982	0.013	0.899	0.016	0.964	0.079
University Traces						
non-Skype	0.93	0.04	0.981	0.122	0.935	0.047
Skype	0.96	0.07	0.878	0.019	0.953	0.065
Italy Traces						
non-Skype	0.0	0.0	0.0	0.091	0.0	0.011
Skype	1.0	0.0	0.909	0.0	0.989	0.0

packet traffic correctly on these networks.

For detecting Skype based on the packet header features, Table 8 lists the results for the three machine learning algorithms on the (University) training, and independent test traces. All models return high detection rate for Skype traffic. Moreover, C4.5 and SBB-GP appear to provide the strongest performance with consistently better FPR and DR. SBB-GP classifier achieves ≈95% DR and ≈7% FPR on University traces and ≈99% DR on the Italy Test traces whereas C4.5 achieves ≈96% DR and ≈7% FPR on the University Test trace and 100% DR Italy Test trace. In this case, robustness results illustrates the success of the packet header based approach in identifying encrypted Skype tunnels based on a single packet's attributes without using IP addresses, port numbers or payload information. It should be noted here that the FP rate for Skype and DR for non-Skype are zero in the Italy traces because this network trace contains only Skype traffic. In the case of classifying Skype VoIP based on packet header feature set, the robustness results illustrate the success of the packet header based approach in identifying encrypted Skype tunnels based on a single packet's attributes without using IP addresses, port numbers or payload information.

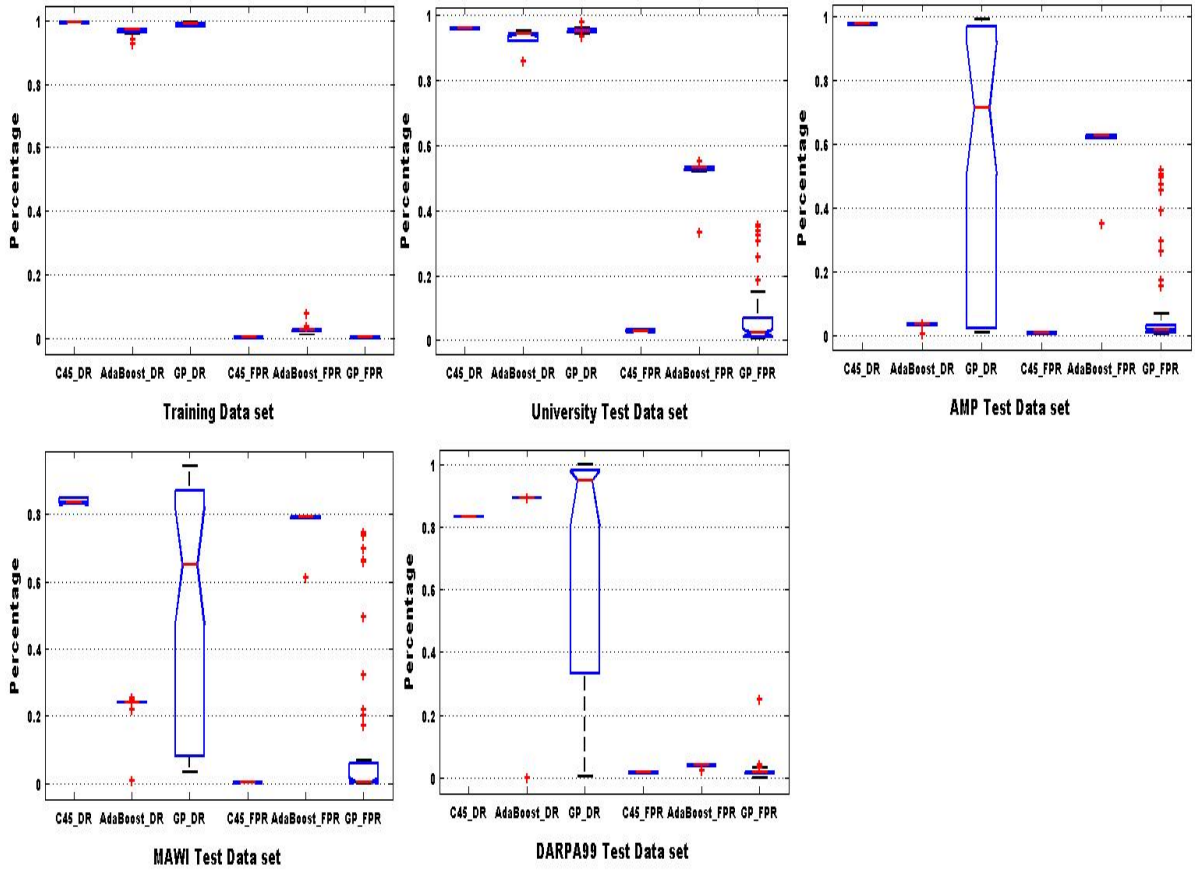


Figure 9: Results on Data sets for Flow based Feature set for SSH detection.

6.2. *Results of flow experiments for SSH and Skype identification*

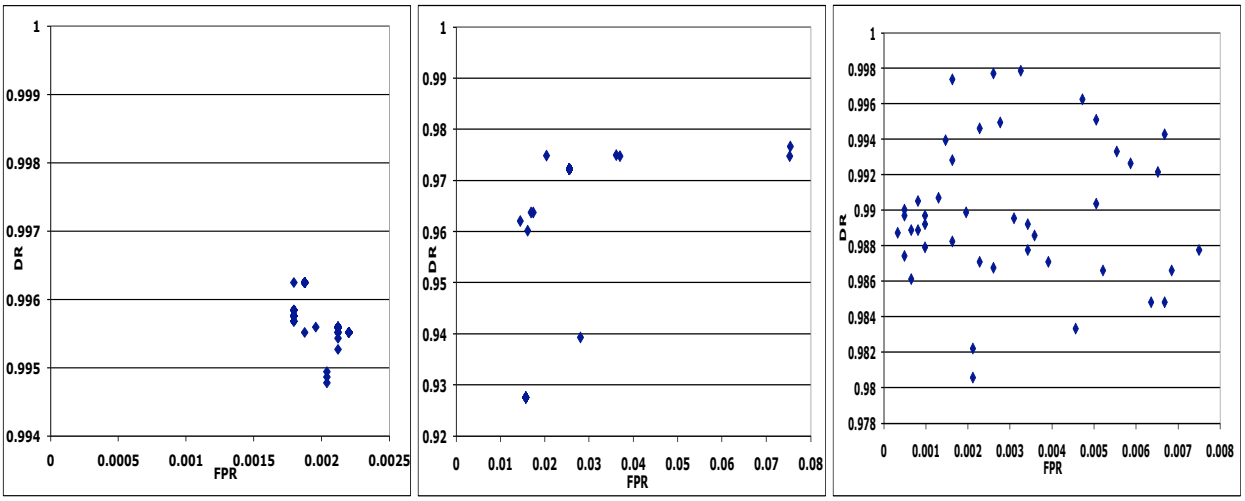
In these set of experiments, the objective is to identify SSH and Skype on a flow by flow basis using only the set of features given in Table 3. Training results presented in this section are also based on 50 runs.

Results given in Fig. 9 shows that C4.5 and GP based classification approaches are much better than AdaBoost algorithm employed in identifying the SSH traffic based on the flow based feature set on the training data set. Moreover, in the case of C4.5 and GP, much lower variance (Table 9) implies that the corresponding solutions generalize to the wider case, implicit in the test results. We use these trained models, on all of the complete traces employed. Furthermore, Table 1 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. Fig. 9 shows the DR and FPR for all 50 models on the test data sets. On average, GP is much better than the other machine learning algorithms on the test data sets in terms of high DR and low FPR. Moreover, these results show that 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 [58]. On the other hand, C4.5 and AdaBoost models/solutions are commonly similar.

To select the best trained model of each machine learning algorithm, we plot the training performance in ROC curve for each of the three machine learning algorithms. Fig. 10 summarizes the solutions. For SSH, there are seven that are non-dominated for GP, three that are non-dominated for AdaBoost and one that is non-dominated for C4.5.

Table 9: Standard Deviation of Results on the training data for SSH detection based on flow feature set.

	C4.5		AdaBoost		GP	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample (subset of university) x 50						
non-SSH	0.0001	0.0004	0.011	0.017	0.002	0.004
SSH	0.0004	0.0001	0.017	0.011	0.004	0.002



(a) C4.5 (b) AdaBoost (c) GP

Figure 10: ROC Curve plot for the three classifiers for training performance using Flow based Feature set for SSH detection (DR versus FPR)

We selected the highest performance solution in terms of high DR and low FPR out of these non-dominated solutions for GP, AdaBoost and C4.5 and then evaluated on the test data sets. Furthermore, Table 1 shows that these traces indeed belong to substantially different networks. Therefore, we believe that only well generalized (robust) models are able to classify SSH traffic correctly on these networks.

Results are summarized in terms of accuracy. Table 10 lists the results for the three machine learning algorithms on the (University) training and 'test (validation)' traces, and independent test traces. That is to say, all model construction takes place on the University Training partition. Post training evaluation is conducted under University Test and AMP, MAWI and DARPA99 traces where none were encountered during training. Naturally, the University Test partition will more closely reflect the training behavior than the AMP, MAWI and DARPA99 test data sets. All models tend to return a marginally better detection rate on out-class data than in-class data on the test data sets, whereas the false positive rates are generally better under the in-class data.

For SSH flows, results show that the SBB-GP classifier performs better than the other classifiers on the majority of the data sets. SBB-GP classifier achieves 98% DR and ≈2% FPR on DARPA99 traces, 98% DR and ≈1% FPR on the AMP traces, and 89% DR and 0.2% FPR on MAWI traces, Table 10. Moreover, the SBB-GP classifier was the most consistent performer across all test and training conditions, while also being competitive with C4.5 under University traces (particularly in terms of SSH false positive rate and SSH detection rate). This not only shows that the model, which the SBB-GP classifier learned during training is robust (generalized) enough to be tested on real world network traces, but also verifies that accurate differentiation between SSH tunnels and non-SSH traffic is possible without employing port numbers, IP addresses and payload information. Last but not the least, these results also demonstrate that to achieve high detection and low false positive rates, temporal information is necessary in case of SSH. We believe this is the most important difference between the flow based features and single packet header based features.

Table 10: Best results out of 50 runs for each Classifier on training and testing data sets for SSH detection based on flow feature set.

	C4.5		AdaBoost		GP	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample						
non-SSH	0.998	0.004	0.980	0.025	1.000	0.010
SSH	0.996	0.002	0.975	0.020	0.990	0.000
University Traces						
non-SSH	0.971	0.040	0.667	0.063	0.964	0.050
SSH	0.960	0.029	0.937	0.333	0.950	0.036
DARPA99 Traces						
non-SSH	0.984	0.166	0.962	0.106	0.983	0.017
SSH	0.834	0.016	0.894	0.038	0.983	0.017
AMP Traces						
non-SSH	0.991	0.027	0.652	0.965	0.992	0.019
SSH	0.973	0.009	0.035	0.348	0.981	0.008
MAWI Traces						
non-SSH	0.995	0.152	0.391	0.760	0.998	0.110
SSH	0.848	0.005	0.240	0.609	0.890	0.002

Table 11: Standard Deviation of Results on the training data for Skype detection based on flow feature set.

	C4.5		AdaBoost		GP	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample (subset of university) x 50						
non-Skype	0.0002	0.0001	0.0046	0.0008	0.026	0.0214
Skype	0.0001	0.0002	0.0008	0.0046	0.0214	0.026

On the other hand, in the case of Skype experiments using flow based features, we first trained each classifier on our training data set using the same feature set. Then, we tested each trained model (C4.5, AdaBoost and SBB-GP) on the test data sets, namely, University traces, and Italy traces. Results presented in Fig. 11 illustrates that C4.5 based classification approach is much better than other algorithms employed in identifying the Skype flow traffic based on the training data set. Moreover, in the case of C4.5, much lower variance (Table 11) implies that the corresponding solutions generalize to the wider case, implicit in the test results. The box plot demonstrates the diversity of performance in terms of DR and FPR for all 50 models on the test data sets for each machine learning algorithm, Fig. 11. On average, C4.5 is much better than other machine learning algorithms on the test data sets in terms of high DR and low FPR.

We plot the trained performance in ROC curve for each of the three machine learning algorithms to determine the best trained model for each machine learning algorithm. Fig. 12 summarizes solutions. For Skype, there are five that are non-dominated for GP, three that are non-dominated for AdaBoost and one that are non-dominated for C4.5. We considered the best performing solution (high DR and low FPR) out of these non-dominated solutions for GP, AdaBoost and C4.5 and then evaluated on the test data sets. Additionally, Table 1 illustrates that these traces indeed belong to significantly different networks. Therefore, we believe that only well generalized models are able to classify Skype flow traffic correctly on these networks.

In the case of Skype flows, Table 12 shows that C4.5 based classification approach is much better than other machine learning algorithms employed in identifying the Skype traffic. In this case, C4.5 based system can correctly classify ≈98% of the instances with less than 1% FPR on University trace and ≈100% DR and 0% FPR on Italy Trace. In these set of experiments SBB-GP based system closely follows the performance of C4.5 based system (with ≈91% DR and ≈5% FPR on University Trace and ≈99% DR and 0% FPR on Italy trace) whereas the AdaBoost based system

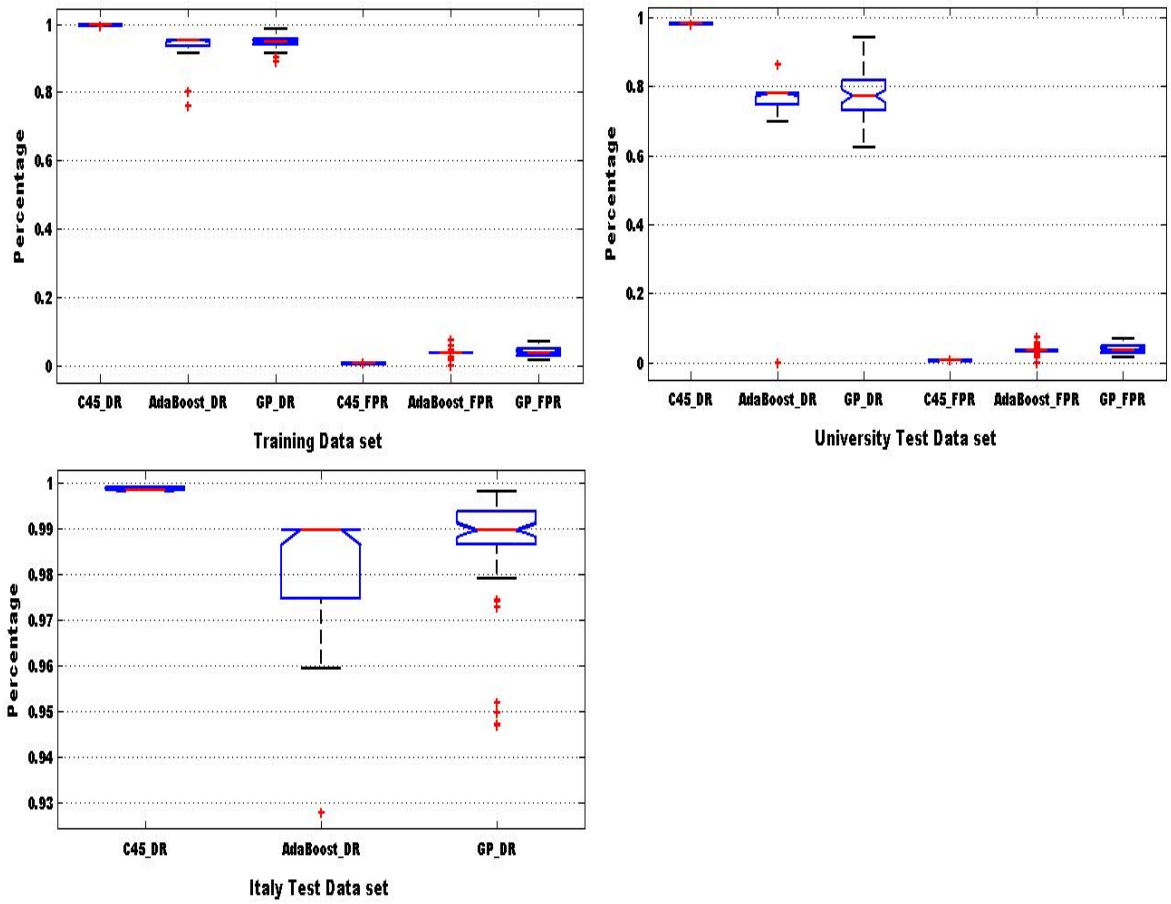


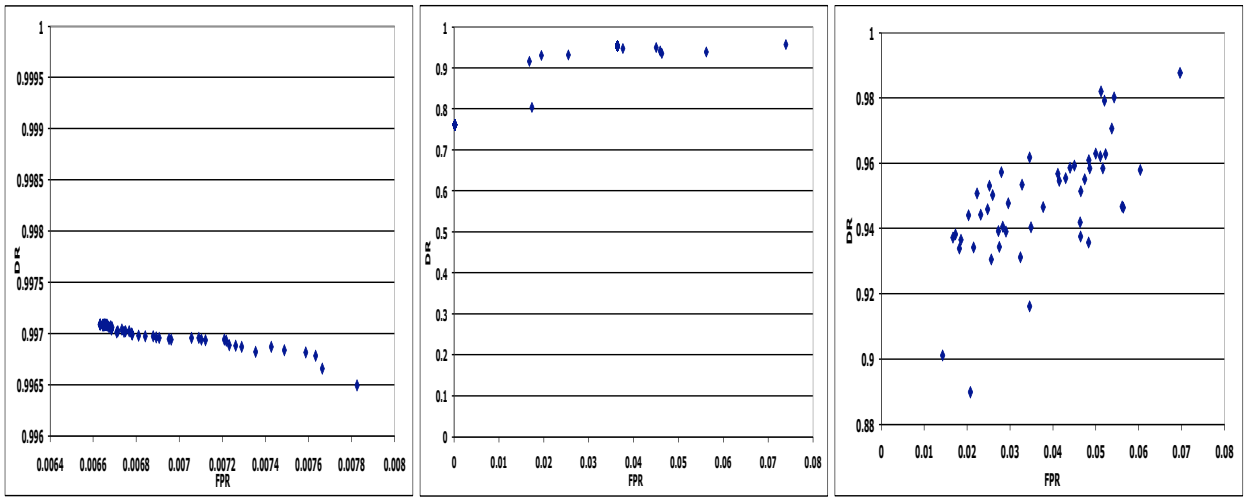
Figure 11: Results on Data sets for Flow based Feature set for Skype detection.

Table 12: Best results out of 50 runs for each Classifier on training and testing data sets for Skype detection based on flow feature set.

	C4.5		AdaBoost		GP	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample						
non-Skype	0.993	0.003	0.964	0.0471	0.95	0.02
Skype	0.997	0.007	0.953	0.036	0.98	0.05
University Traces						
non-Skype	0.993	0.0182	0.964	0.22	0.948	0.086
Skype	0.982	0.007	0.78	0.036	0.914	0.052
Italy Traces						
non-Skype	0.0	0.001	0.0	0.01	0.0	0.006
Skype	0.999	0.0	0.99	0.0	0.994	0.0

performs the poorest of the three.

These results demonstrate that the model, which the SBB-GP and C4.5 classifiers learned during training is robust (generalized) enough to be tested on real world network traces, but also verifies that accurate differentiation between



(a) C4.5 (b) AdaBoost (c) GP

Figure 12: ROC Curve plot for the three classifiers for training performance using Flow based Feature set for Skype detection (DR versus FPR)

SSH/Skype tunnels and non-SSH/non-Skype tunnels is possible without employing port numbers, IP addresses and payload information.Last but not the least, these results also demonstrate that to achieve high detection and low false positive rates, temporal information is necessary in case of SSH/Skype. We believe this is the most important difference between the flow based features and single packet header based features. Furthermore, it is possible to have a generic attribute set that can be employed to identify encrypted tunnels such as SSH and Skype.

6.3. Classifying applications running over SSH: multi-class classification – all applications

In this case, NIMS data set is labeled into multi-classes depending on SSH services/classes (SHELL, SCP, SFTP, X11, Local and Remote tunneling) and background traffic (FTP, TELNET, DNS, HTTP and P2P limewire). The applications in each data set are separated into 2 classes in-class (SSH applications) and out-class (background). The in-class contains 6000 flows (1000 flows for each application) while the out-class contains 5000 flows (1000 flows for each application), in total 11,000 flows. The NIMS test data set consists of the following applications Local and Remote tunneling, SCP, SFTP, X11, SHELL, TELNET, FTP, HTTP, DNS and P2P limewire. Each of them contains the following number of flows 2557, 2422, 2444, 2412, 2355, 2491, 1251, 1728, 11,904, 38,016, and 646,271, respectively. Thus, NIMS test data set consists of 713,851 flows in total. In this case, results show that SBB-GP performs much better than AdaBoost and C4.5 algorithms in classifying applications running over SSH, Table 13. SBB-GP can correctly classify ≈99% of instances with a very low false positive rate [roughly 0% FPR].

6.4. Classifying applications running over SSH: multi-class classification – only SSH applications

In this case, NIMS data set is labeled into multi-classes depending on SSH services/classes (SHELL, SCP, SFTP, X11, Local and Remote tunneling) and background traffic (FTP, TELNET, DNS, HTTP and P2P limewire). The applications in each data set are separated into two classes in-class (SSH applications) and out-class (background). The in-class contains 6000 flows (1000 flows for each application). The NIMS test data set consists of the following applications Local and Remote tunneling, SCP, SFTP, X11 and SHELL. Each of them contains the following number of flows 2557, 2422, 2444, 2412, 2355, and 2491, respectively. Thus, in this case, NIMS test data set consists of 14,681 flows in total. Table 14 shows again, that SBB-GP outperforms C4.5 and AdaBoost algorithms in classifying applications running over SSH. SBB-GP can correctly classify ≈99% of the instances with a very low false positive rate [roughly 0% FPR].

Table 13: Results on the NIMS data - Multi-class – All applications

	C4.5		AdaBoost		GP	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Results on Training data sets 10 fold-cross validation						
LT	1.0	0.0	0.605	0.0	0.998	0.0006
RT	1.0	0.0	0.0	0.0	0.979	0.0
SCP	0.992	0.001	0.0	0.0	0.926	0.0
SFTP	1.0	0.0	0.0	0.0	0.993	0.0002
X11	0.989	0.001	0.0	0.0	0.987	0.0
SHELL	0.998	0.0	0.998	0.295	1.0	0.0006
TELNET	0.998	0.0	0.0	0.0	0.992	0.0
FTP	0.993	0.001	0.0	0.0	0.997	0.01
HTTP	0.993	0.0	0.0	0.0	0.772	0.003
DNS	0.99	0.0	0.0	0.0	0.913	0.643
P2P	0.988	0.001	0.0	0.0	0.316	0.07
Results of All applications on Testing data						
LT	1.0	0.0	0.986	0.0	0.998	0.0006
RT	1.0	0.0	0.0	0.0	0.979	0.0
SCP	0.828	0.006	0.0	0.0	0.926	0.0
SFTP	0.967	0.264	0.0	0.0	0.993	0.0002
X11	0.714	0.033	0.0	0.0	0.987	0.0
SHELL	0.997	0.0	0.997	0.01	1.0	0.0006
TELNET	0.998	0.0	0.0	0.0	0.992	0.0
FTP	0.995	0.0	0.0	0.0	0.997	0.01
HTTP	0.991	0.0	0.0	0.0	0.772	0.003
DNS	1	0.0	0.0	0.0	0.913	0.643
P2P	0.995	0.001	0.0	0.0	0.316	0.07

LT=local tunneling, RT= remote tunneling, P2P= P2P(lime-wire)

6.5. Sensitivity to configuration parameters

In this section, we discuss the effect of the parameters on the performance of the C4.5 and the SBB-GP based classifiers, since they are the top two performers in our experiments. In this case, we have used the default Weka parameters (there are 11 parameters in Weka) for the C4.5 based classifier, because our aim is not to find the best parameter set, but to investigate whether such a classifier will work out of the box and what its performance would be under such circumstances. The most important parameter to 'tune' under C4.5 is the pruning/confidence factor, where this has a direct impact on resulting model complexity, and model complexity is potentially related to generalization performance/real-time operation. Fig. 13 summarizes the impact of varying the confidence factor (CF) on detection rate (DR), false positive rate (FPR) and rule count.[4] Essentially as the CF increases (less pruning) the resulting C4.5 model becomes more specific. The FPR decreases and rule count increase; however, in this particular data set there is no change in DR. The best DR and FPR appear in the interval of CF factors of 0.3–0.35. After this any further improvement to FPR is negligible. Unfortunately, this also corresponds to the higher rule counts under C4.5 (400 rules). This sweet spot in configuration also happens to correspond to the default Weka parameterization for C4.5.

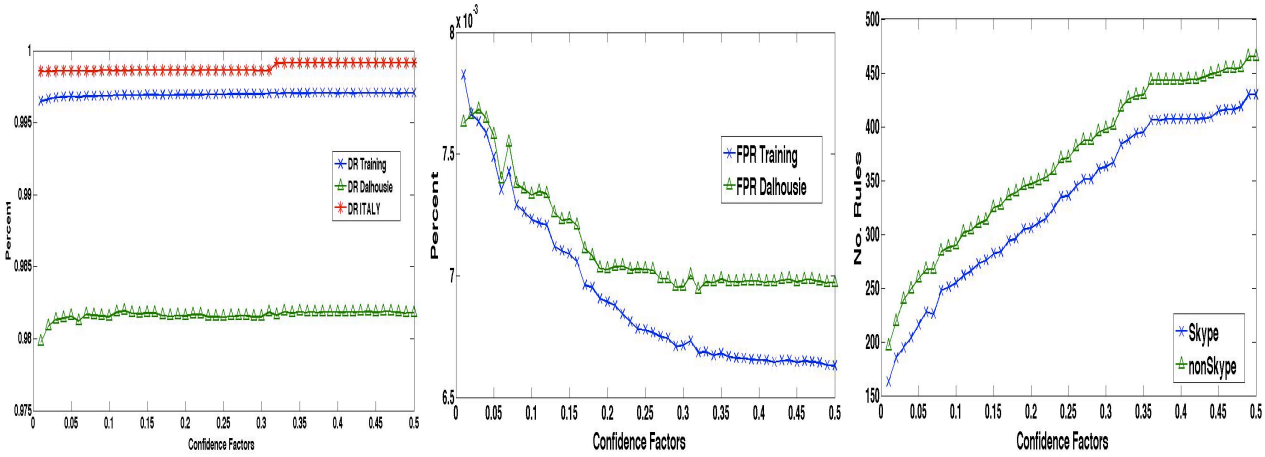
The principle property of variation in models of evolutionary computation are the seed parameters used to initialize the various stochastic processes behind population initialization and search operators. This source of variation is addressed through the use of multiple runs (50 in this case) and is reported throughout the aforementioned experimental study. Secondary parameters might include the number of generations, population size, team size, program length, frequency of applying search operators and total number of registers. Of these, team size and program length are free

[4]Skype detection task and flow features.

Table 14: Results on the NIMS data sets - Multi-class – Only SSH applications

	C4.5		AdaBoost		GP	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Results on Training data sets 10 fold-cross validation						
LT	0.998	0.0	0.998	0.592	0.998	0.0
RT	0.996	0.0	0.0	0.0	0.998	0.0
SCP	0.993	0.002	0.0	0.0	1.0	0.0004
SFTP	0.993	0.001	0.0	0.0	0.996	0.0
X11	0.992	0.001	1.0	0.208	0.986	0.0
SHELL	0.99	0.003	0.0	0.0	1	0.006
Results of All applications on Testing data						
LT	0.0	0.209	0.979	0.597	0.988	0.0003
RT	0.0	0.199	0.0	0.0	0.978	0.0
SCP	0.995	0.002	0.0	0.0	0.993	0.002
SFTP	0.99	0.0	0.0	0.0	0.989	0.002
X11	0.003	0.193	1.0	0.205	0.989	0.0006
SHELL	0.003	0.202	0.0	0.0	0.998	0.008

LT=local tunneling, RT= remote tunneling, P2P= P2P(lime-wire)



(a) DR (b) FPR (c) Rules

Figure 13: Sensitivity of C4.5 for changing the confidence factor parameter.

to adapt. Thus as long as runs do not approach the team size and program length limits, the choice of such parameters is independent of the parameterization; as is the case here. Any generational limit is generally fixed to reflect the computational overhead of the task at hand. That is to say, as long as sufficient evaluations are performed to reach a performance plateau, the value of considering further evaluations need to be traded off against a law of diminishing returns. Likewise is true regarding the size of the point population used to sample from the wider training set. The larger the point population the more significant the computational overhead in performing any single fitness evaluation; particularly under Pareto formulations as used here. Conversely the smaller the point population the greater the sensitivity to any single sample from the point population. Previous research has also demonstrated that even when there is an order of magnitude difference in point population size, there is little impact on the quality of evolved solutions [59]. The remaining parameters have been studied extensively by the linear GP literature. In particular the work of Brameier and Banzhaf identifies Max. Register Count as the single most significant parameter on linear GP;

Table 15: Features used by All Classifiers for in/out-class for SSH Packet Header Feature

	GP		C4.5		AdaBoost	
	in-class	**out-class**	**in-class**	**out-class**	**in-class**	**out-class**
1	frame.len	frame.len	frame.cap_len	frame.cap_len	frame.cap_len	frame.cap_len
2	ip.len	frame.cap_len	frame.pkt_len	frame.pkt_len	frame.len	frame.len
3	ip.flags	ip.flags	frame.time_delta	frame.time_delta	ip.ttl	ip.ttl
4	ip.flags.rb	ip.frag_offset	ip.len	ip.len	tcp.flags	tcp.flags
5	ip.frag_offset	ip.ttl	ip.ttl	ip.ttl	tcp.flags.reset	tcp.flags.reset
6	tcp.seq	ip.proto	ip.flags	ip.flags		
7	tcp.ack	ip.checksum_good	tcp.len	tcp.len		
8	tcp.hdr_len	tcp.seq	tcp.ack	tcp.ack		
9	tcp.flags.ecn	tcp.ack	tcp.seq	tcp.seq		
10	udp.checksum	tcp.hdr_len	tcp.nxtseq	tcp.nxtseq		
11		tcp.flags.urg	tcp.flags	tcp.flags		
12		tcp.flags.fin	tcp.flags.reset	tcp.flags.reset		
13		tcp.checksum_bad	tcp.flags.push	tcp.flags.push		
14		udp.checksum	tcp.flags.fin	tcp.flags.fin		
15		udp.checksum_bad	tcp.flags.syn	tcp.flags.syn		
16			tcp.window_size	tcp.window_size		

Chapter 7 in [60]. Such a result is independent of problem domain as it reflects the ratio of registers to instructions per program. In short, the SBB algorithm adapts program length and team size to the problem domain at hand. Changes to point population size and team population size have little impact beyond some nominal figure (say 50 individuals) and, in the limit, are sensitive to the total computational cost of performing a run. Likewise a generational limit is imposed, such that performance plateaus before a run is considered to have completed execution.

7. Discussion

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

7.1. Analysis for the packet header based approach for SSH and Skype

For SSH/Skype packet header solutions, Tables 15 and 16 summarize these findings for the case of AdaBoost, SBB-GP and C4.5, respectively. AdaBoost clearly uses the lowest number of attributes relative to SBB-GP and C4.5. for SSH/Skype detection. However, we will focus on SBB-GP since it has the second lowest attributes utilization and on most of the experiments achieved the highest performance results in terms of high DR and low FPR among the three learning algorithms. 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-GP is the only model to make use of '*ip.flags.rb,ip.fraq_offset, tcp.hdr_len and tcp.flags.ecn*' for SSH detection, Table 15. Moreover, SBB-GP is the only model to focus on attributes based on the header length for the frame header, IP header and TCP header for Skype detection, Table 16. These attributes 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 4 of 10 attributes shared between C4.5 and SBB-GP under SSH detection and 6 of 15 attributes under non-SSH detection (w.r.t. SBB-GP attribute selection) and only 6 of 13 attributes shared between C4.5 and SBB-GP under Skype detection and 8 of 14 attributes under non-Skype detection (w.r.t. SBB-GP attribute selection). What is certainly clear, however, is that a significant degree of preference exists

```
R[3] <- R[3] I[20] diff
R[0] <- R[0] R[3] diff
R[3] <- R[3] R[0] cos
R[0] <- R[0] I[30] exp
R[3] <- R[3] R[0] mod
R[3] <- R[3] I[0] div
R[0] <- R[0] I[1] log
R[1] <- R[1] R[0] exp
R[3] <- R[3] R[1] sum
R[0] <- R[0] R[3] exp
R[0] <- R[0] I[2] prod
```

Figure 14: SBB-GP solution for Skype based on packet header feature set

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 for SSH packet header solutions, we note that AdaBoost generates 10 signatures for SSH traffic and 9 signatures for non-SSH traffic; whereas C4.5 employs 284 signatures for SSH classification and 342 signatures to classify non-SSH traffic. Conversely, SBB-GP uses 2 individuals for SSH classification and 10 for non-SSH. We also note that instruction counts for the SSH classifying individuals was 15 and 25 instructions, respectively; whereas individuals engaged in classifying non-SSH utilized 5 (two off), 6, 8, 10, 11, 12, 13 and 16 (two off) instructions. In terms of solution complexity for Skype packet header solutions, we note that AdaBoost generates 10 signatures for Skype traffic and 8 signatures for non-Skype traffic; whereas C4.5 employs 966 signatures for Skype classification and 1022 signatures to classify non-Skype traffic. Conversely, SBB-GP uses 6 individuals for Skype classification and 7 for non-Skype, Fig. 14. We also note that instruction counts for the Skype classifying individuals was 2 and 11, 13, 14 (two off) and 15 instructions respectively; whereas individuals engaged in classifying non-Skype utilized 8, 10 (two off), 12, 14, and 15 (two off) instructions. In short, the simplicity of SBB-GP solutions does not appear to be traded off for classifier complexity, in effect emphasizing the significance of support for problem decomposition in this application. Moreover, the SBB-GP solution for Skype based on packet header feature can process ≈13,210 packets per second on a MacBook 2 GHz Intel Core 2 Duo with 2 GB RAM in terms of classifying off-line data sets.

7.2. Analysis for the flow based approach for SSH and Skype

For SSH flow solutions, again, AdaBoost uses a lower total count of attributes both for SSH (in-class) and non-SSH (out-class) relative to either SBB-GP or C4.5, and lower counts of attributes for SSH tunnel detection. Conversely, C4.5 uses the largest set of attributes as a whole or class-wise. We focus our analysis for SBB-GP since it achieved the best result for classifying SSH encrypted traffic. In this case, SBB-GP is the only model to focus on the inter-arrival time and packet length of the forward direction (8 attributes) and packet length of the backward direction – intuitively, this makes sense since SSH applications are mostly interactive applications (user–machine), Fig. 15. These features may give more insight on the behavior of the user and can provide more information to predict what the payload might be. Such an indication for network/system administrators can be very useful since encrypted content can hide the detection of anomalous activities that can harm the system or steal sensitive data. Also of interest is the low level of overlap in shared attributes, with only 13 of 16 attributes shared between C4.5 and SBB-GP under SSH detection. Again, it is clear that a significant degree of preference exists in attribute selection relative to the machine learning model. In terms of solution complexity, AdaBoost generates 10 signatures for SSH traffic and 10 signatures for non-SSH traffic; whereas C4.5 employs 16 signatures for SSH classification and 25 signatures to classify non-SSH traffic. Conversely, SBB-GP uses 14 individuals for SSH classification and 14 for non-SSH. In short, the simplicity of SBB solutions does not appear to be traded off for classifier complexity while still achieving very good accuracy, in effect emphasizing the significance of support for problem decomposition in this application.

Table 16: Features used by All Classifiers for in/out-class for Skype Packet Header Feature Set

	GP		C4.5		AdaBoost	
	in-class	**out-class**	**in-class**	**out-class**	**in-class**	**out-class**
1	frame.time_delta	frame.time_delta	frame.time_delta	frame.time_delta	frame.time_delta	frame.time_delta
2	frame.pkt_len	frame.pkt_len	frame.pkt_len	frame.pkt_len	ip.checksum	ip.checksum
3	frame.cap_len	frame.cap_len	frame.cap_len	frame.cap_len	ip.len	ip.len
4	frame.len	frame.len	ip.ttl	ip.ttl	ip.ttl	ip.ttl
5	ip.len	frame.marked	ip.len	ip.len	tcp.checksum	tcp.checksum
6	ip.flags.rb	ip.flags	ip.flags	ip.flags		
7	ip.flags.df	ip.len	ip.proto	ip.proto		
8	ip.ttl	ip.checksum_good	ip.flags.mf	ip.flags.mf		
9	ip.checksum_bad	tcp.len	ip.checksum	ip.checksum		
10	tcp.hdr_len	tcp.seq	tcp.ack	tcp.ack		
11	tcp.window_size	tcp.hdr_len	tcp.window_size	tcp.window_size		
12	tcp.checksum	tcp.flags.cwr	tcp.seq	tcp.seq		
13	udp.checksum_bad	tcp.flags.ack	tcp.nxtseq	tcp.nxtseq		
14		tcp.window_size	tcp.len	tcp.len		
15			tcp.flags	tcp.flags		
16			tcp.flags.fin	tcp.flags.fin		
17			tcp.flags.push	tcp.flags.push		
18			tcp.flags.syn	tcp.flags.syn		
19			tcp.flags.reset	tcp.flags.reset		
20			udp.length	udp.length		

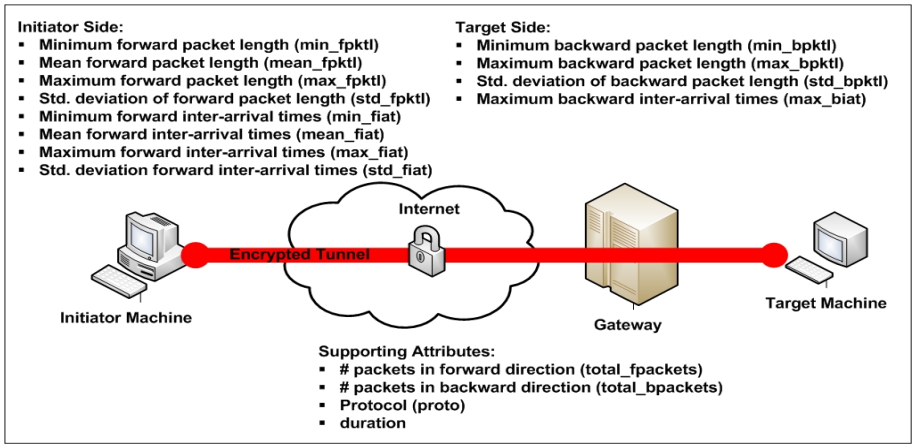


Figure 15: The selected attributes by SBB-GP for SSH

On the other hand, for Skype, Figs. 17 and 18 summarize the results for the case of SBB-GP and C4.5, respectively. SBB-GP clearly uses a lower count of attributes for Skype 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-GP chooses the Protocol attribute since Skype application runs on both TCP and UDP protocols; while C4.5 utilizes the attributes based on packet size and inter-arrival time, Fig. 18. Moreover, again, the overlap in shared attributes among the machine learning models is low, with only 2 of 5 attributes shared between C4.5 and SBB-GP under Skype detection. Again, it is clear that a significant degree of preference exists in attribute selection relative to the machine

```
R[7] <- R[7] R[5] cos
R[3] <- R[3] R[7] exp
R[2] <- R[2] I[4] log
R[3] <- R[3] R[2] sum
R[0] <- R[0] I[15] exp
R[0] <- R[0] R[3] cos
```

Figure 16: SBB-GP solution for Skype based on flow feature set

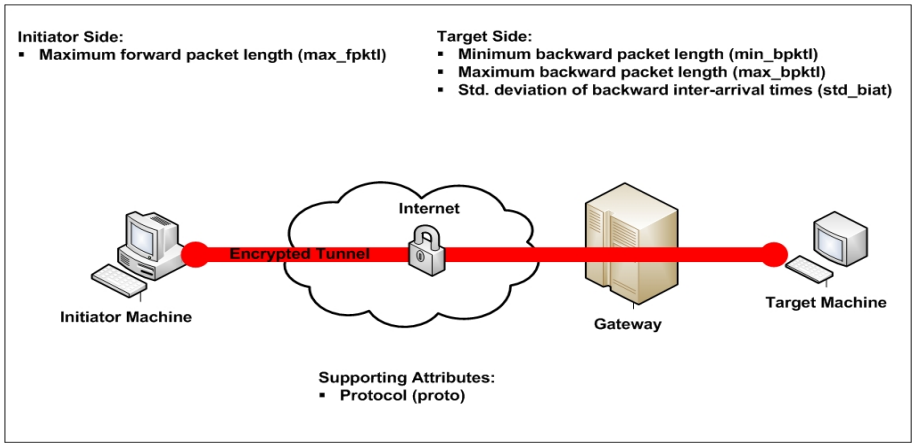


Figure 17: The selected attributes for SBB-GP for Skype

learning model; thus, attempting to provide a limited 'hand crafted' set of attributes for Skype identification is likely to be counter-productive. In terms of solution complexity, we note that AdaBoost generates 10 signatures for Skype traffic and 7 signatures for non-Skype traffic; whereas C4.5 employs 133 rules for identifying Skype traffic and 135 signatures for non-Skype traffic. Conversely, SBB-GP uses 2 individuals for Skype classification and 10 for non-Skype. Similar to the SSH case, the simplicity of SBB-GP solutions does not appear to be traded off for classifier complexity in the identification of Skype tunnels either. Fig. 16 shows the SBB-GP solution for detection Skype traffic based on flow feature set. The SBB-GP model takes ≈50 μs (≈20,012 flow records per second) to process one flow record on a MacBook 2 GHz Intel Core 2 Duo with 2 GB RAM.

In summary, machine learning algorithms such as AdaBoost, SBB-GP and C4.5 select the most appropriate attributes (among the set given) to build their classifier model. We used this information to distinguish 16 flow attributes from a set of 22 that are used in our experiments for identifying SSH tunnel using C4.5 whereas 18 flow attributes (again, out of 22) using C4.5 to identify Skype tunnels. On the other hand, if SBB-GP is chosen as the preferred model to generate automatic signatures then 16 flow attributes and 5 flow attributes are necessary to identify SSH and Skype tunnels, respectively (again out of the same 22). Figs. 15, 17 and 18.

Looking at the solution/model of SBB-GP and C4.5 for SSH and Skype flow based feature set, some of the overlapping attributes might shed some light on the relationship of the encryption and the integrity of packets from one angle and ensuring the quality of service while maintaining strong security as another angle. The SBB-GP

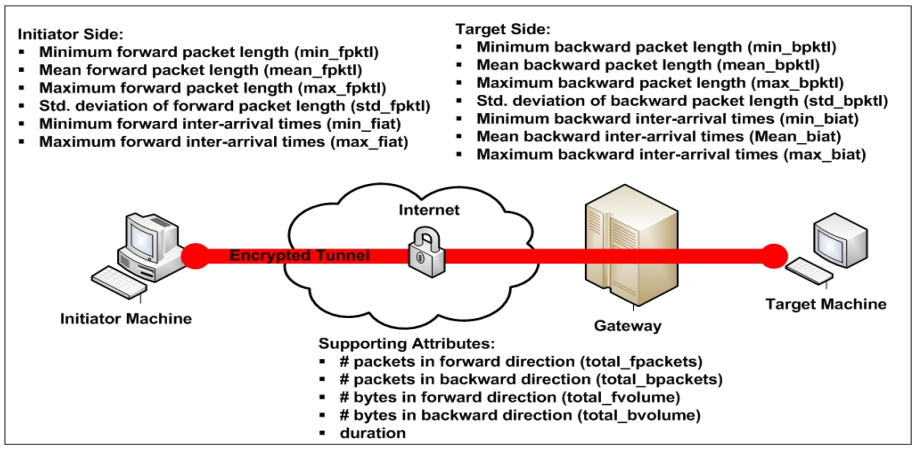


Figure 18: The selected attributes for C4.5 for Skype

models for SSH and Skype contain four attributes that are common for Skype and SSH identification whereas the C4.5 models/solution for SSH and Skype include 12 attributes that are shared between the models/solutions for Skype and SSH classification. The attributes for SSB-GP are: max_fpktl, min_bpktl, max_bpktl and protocol; while the attributes for C4.5 are: max_bpktl, std_bpktl, total_fvolume, total_fpackets, min_fpktl, max_fiat, min_biat, min_bpktl, mean_fpktl, mean_bpktl, std_fpktl and max_fpktl.

Intuitively, what these algorithms learned from the data makes sense. In order to correctly identify SSH/Skype traffic, the classifier naturally needs to explore both forward and backward directions of the traffic. Each direction has its unique signature given that an Initiator machine (starts connection) and a Corresponder machine (responds to the connection) operate differently. Thus, we believe that the features listed in Figs. 15, 17 and 18 are what the learning model used to discover the encrypted tunnel. These features are separated into two: (i) attributes from Initiator to Corresponder (Forward direction); and (ii) attributes from Corresponder to Initiator (Backward direction). The attributes in the forward direction are also based on the packet length and inter-arrival time since the forward direction depends on the initiator requesting information (e.g. min_fpktl, mean _fpktl, max_fpktl, std_fpktl, min_fiat, mean_fiat, max_fiat and std_fiat). On the other hand, the attributes in the backward direction are based on the packet length and the inter-arrival time since the backward direction is based on the Corresponder side responding to the Initiator requests (min_bpktl, mean _bpktl, max_bpktl, std_bpktl, min_biat, mean_biat, max_biat and std_biat).

For example, the minimum length for a packet is in part effected by the length of the request made by the Initiator. On the other hand, the standard deviation for a packet length is a measurement of variation of requests, i.e. different commands used by the Initiator. In other words, the standard deviation measures the spread of packet length, which can indicate different commands Initiators run. Consequently, the minimum and standard deviation of packet length measurements can shed some light on the behavior of the Initiator in choosing commands to do the work and can provide more information to predict what the payload might be. Such an indication can be very useful for network/system administrators since encrypted content can hide the detection of anomalous activities that can harm the system or steal sensitive data.

7.3. Limitations

Indeed, no signature based method in traffic classification can be perfect, given that there can always be some new (unseen) application. Thus, the major challenge to traffic classification in general is evasion. All classification methods can be evaded. For example, payload-based approach can be evaded by encrypting the packet payload, and port-based approach can be evaded by dynamically changing port numbers. On the other hand, approaches based on flow statistics using packet size and inter-arrival time attributes are sensitive to altering these attributes. If the attackers want to evade the proposed method, they can modify the size of the packets in the entire connection by randomly padding packet payloads. The accuracy of the proposed method might be decreased if those features that depend on

packet size were modified from the application behavior. However, it is not that easy to obfuscate application behavior without presenting large amount of overhead. Another limitation of any classification system is obtaining (generating) the training data set. The generality and accuracy of the classifier depends on the quality of the training data sets. A meaningful and representative training data set is hard to find and generating one is resource and time consuming. Moreover, since the classifier generates the signatures automatically from the training data set, the accuracy of the classifier might decrease if the signatures/models from the trained classifiers are applied to network traffic that have different characteristics or behavior (such as new applications are developed or old applications change their behavior). Indeed, in such cases, the signatures/models need to be updated by retraining the classifiers. That is why we believe it is very important to conduct robustness analysis on such classifiers.

8. Conclusions

In this work, we investigate the robustness of the models/signatures automatically generated by AdaBoost, GP and C4.5 learning algorithms for distinguishing encrypted tunnels (namely SSH and Skype) from non-encrypted tunnels in a given traffic trace. To do so, we employ public traffic traces from DARPA99, AMP, MAWI and Italy repositories based on the previous research as well as employing traffic traces captured on our Dalhousie Campus network. We evaluate the aforementioned learning algorithms using packet header based features and traffic flow based features.

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.

Furthermore, the classification based approach can be employed with either using packet header only attributes or flow attributes. We have investigated both attribute sets (feature sets) on the traces employed and shown that flow based features performed statistically significantly better than the packet header based features. It should be noted here that the classification based approach does not use IP addresses, port numbers and payload data.

In all cases, SBB-GP based classification approach performed better than the other learning algorithm based methods. In these experiments, in the worst case scenario, the GP based classifier can achieve a 89% DR and 0.2% 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, the GP based classifier can achieve up to 98% 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 signatures, i.e. solutions, generated by the classification based system are robust generic solutions as well as being easy to understand. Furthermore, we were able to identify services running over SSH such as interactive login sessions (SHELL), tunneling (both local and remote), SCP (secure copy), SFTP (secure file transfer) and X11 activities with high detection rate and low false positive rate, too.

Future work will follow similar lines to perform more tests on different and/or larger data sets in order to continue to evaluate the robustness of the classifiers. 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. Last but not the least, we also want to investigate using both packet header based feature set and flow based feature set side by side. The combination of both feature sets can allow us to classify encrypted traffic in real-time since we can generate packets/flows on-demand from a stream of data. Furthermore, the gain of combining both feature sets can allow us a coarse-grained parallelism, which can increase the accuracy of detecting encrypted traffic.

Acknowledgment

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

References

[1] Internet assigned numbers authority (IANA), http://www.iana.org/assignments/port-number (last accessed October, 2009).
[2] A. W. Moore, K. Papagiannaki, Toward the accurate identification of network applications, in: Passive and Active Network Measurement: Proceedings of the Passive &Active Measurement Workshop, 2005, pp. 41–54.
[3] A. Madhukar, C. Williamson, A longitudinal study of p2p traffic classification, in: MASCOTS '06: Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation, IEEE Computer Society, Washington, DC, USA, 2006, pp. 179–188. doi:http://dx.doi.org/10.1109/MASCOTS.2006.6.
[4] S. Sen, O. Spatscheck, D. Wang, Accurate, scalable in-network identification of p2p traffic using application signatures, in: WWW '04: Proceedings of the 13th international conference on World Wide Web, ACM, New York, NY, USA, 2004, pp. 512–521. doi:http://doi.acm.org/10.1145/988672.988742.
[5] SSH, http://www.rfc-archive.org/getrfc.php?rfc=4251.
[6] Skype, http://www.skype.com/useskype/.
[7] R. Alshammari, A. N. Zincir-Heywood, A flow based approach for ssh traffic detection, Proceedings of the IEEE International Conference on System, Man and Cybernetics - SMC'2007.
[8] R. Alshammari, 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, IEEE Computer Society, Washington, DC, USA, 2008, pp. 156–166.
[9] R. Alshammari, N. Zincir-Heywood, Generalization of signatures for ssh encrypted traffic identification, in: Computational Intelligence in Cyber Security, 2009. CICS '09. IEEE Symposium on, 2009, pp. 167–174.
[10] J. Early, C. Brodley, C. Rosenberg, Behavioral authentication of server flows, in: Proceedings of the 19th Annual Computer Security Applications Conference, 2003, pp. 46–55.
[11] P. Haffner, S. Sen, O. Spatscheck, D. Wang, ACAS: automated construction of application signatures, in: MineNet '05: Proceeding of the 2005 ACM SIGCOMM workshop on Mining network data, ACM Press, New York, NY, USA, 2005, pp. 197–202.
[12] A. D. Montigny-Leboeuf, Flow Attributes For Use In Traffic Characterization, CRC Technical Note No. CRC-TN-2005-003.
[13] A. W. Moore, 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, ACM Press, New York, NY, USA, 2005, pp. 50–60.
[14] N. Williams, S. Zander, G. Armitage, A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification, SIGCOMM Comput. Commun. Rev. 36 (5) (2006) 5–16.
[15] C. Wright, F. Monrose, 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, ACM Press, New York, NY, USA, 2004, pp. 9–15.
[16] T. Karagiannis, K. Papagiannaki, 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, ACM Press, New York, NY, USA, 2005, pp. 229–240.
[17] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, K. Salamatian, Traffic classification on the fly, SIGCOMM Comput. Commun. Rev. 36 (2) (2006) 23–26.
[18] J. Erman, M. Arlitt, A. Mahanti, Traffic classification using clustering algorithms, in: MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data, ACM Press, New York, NY, USA, 2006, pp. 281–286.
[19] SSH FAQ, http://www.rz.uni-karlsruhe.de/ ig25/ssh-faq/.
[20] D. J. Barett, R. E. Silverman, SSH, The Secure Shell: The Definitive Guide, 1st Edition, O'Reilly, 2001.
[21] RFC4254, http://tools.ietf.org/html/rfc4254.
[22] RFC4252, http://tools.ietf.org/html/rfc4252.
[23] RFC4253, http://tools.ietf.org/html/rfc4253.
[24] F. Dijkstra, A. Friedl, et al., Specification of advanced features for a multi-domain monitoring infrastructure, http://www.geant.net/Media_Centre/Media_Library/Pages/Deliverables.aspx (February 2010).
[25] S. A. Baset, H. G. Schulzrinne, An analysis of the skype peer-to-peer internet telephony protocol, in: INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, 2006, pp. 1–11.
[26] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, D. Rossi, Tracking down skype traffic, in: INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, 2008, pp. 261–265.
[27] Y. Zhang, V. Paxson, Detecting back doors, in: In Proceedings of the 9th USENIX Security Symposium, ACM Press, 2000, pp. 157–170.
[28] C. V. Wright, F. Monrose, G. M. Masson, On inferring application protocol behaviors in encrypted network traffic, J. Mach. Learn. Res. 7 (2006) 2745–2769.
[29] L. Bernaille, R. Teixeira, Early recognition of encrypted applications, Passive and Active Measurement Conference (PAM), Louvain-la-neuve, Belgium.
[30] W. Li, M. Canini, A. W. Moore, R. Bolla, Efficient application identification and the temporal and spatial stability of classification schema, Comput. Netw. 53 (6) (2009) 790–809.

[31] F. Palmieri, U. Fiore, A nonlinear, recurrence-based approach to traffic classification, Comput. Netw. 53 (6) (2009) 761–773.
[32] Y. Hu, D.-M. Chiu, J. C. S. Lui, Profiling and identification of p2p traffic, Comput. Netw. 53 (6) (2009) 849–863.
[33] Skype reaches 10 million concurrent users, http://seekingalpha.com/article/50328-ebay-watch-59-earnings-growth-skype-reaches-10-million-concurrent-users (last accessed May, 2010).
[34] D. K. Suh, D. R. Figueiredo, J. Kurose, 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.
[35] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, P. Tofanelli, Revealing skype traffic: when randomness plays with you, SIGCOMM Comput. Commun. Rev. 37 (4) (2007) 37–48.
[36] R. Alshammari, A. N. Zincir-Heywood, Machine learning based encrypted traffic classification: Identifying ssh and skype, in: Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on, 2009, pp. 1–8. doi:10.1109/CISDA.2009.5356534.
[37] R. Alshammari, P. I. Lichodzijewski, M. Heywood, A. N. Zincir-Heywood, Classifying ssh encrypted traffic with minimum packet header features using genetic programming, in: GECCO '09: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, 2009, pp. 2539–2546.
[38] T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, Communications Surveys Tutorials, IEEE 10 (4) (2008) 56 –76. doi:10.1109/SURV.2008.080406.
[39] R. Alshammari, 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 (2008) 203–210.
[40] R. Alshammari, A. N. Zincir-Heywood, A. A. Farrag, Performance comparison of four rule sets: An example for encrypted traffic classification, Privacy, Security, Trust and the Management of e-Business, World Congress on 0 (2009) 21–28.
[41] E. Alpaydin, Introduction to Machine Learning, MIT Press, 2004.
[42] Y. Freund, R. E. Schapire, A short introduction to boosting, Journal of Japanese Society for Artificial Intelligence 14 (5) (1999) 771–780.
[43] P. Lichodzijewski, M. I. Heywood, Managing team-based problem solving with Symbiotic Bid-based Genetic Programming, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2008, pp. 363–370.
[44] E. de Jong, A monotonic archive for pareto-coevolution, Evolutionary Computation 15 (1) (2007) 61–93.
[45] J. Doucette, M. Heywood, Gp Classification under Imbalanced Data Sets: Active Sub-sampling and AUC Approximation, in: European Conference on Genetic Programming, Vol. 4971 of Lecture Notes in Computer Science, 2008, pp. 266–277.
[46] C. D. Rosin, R. K. Belew, New methods for competitive coevolution, Evol. Comput. 5 (1) (1997) 1–29.
[47] NLANR, http://pma.nlanr.net/special.
[48] MAWI, http://tracer.csl.sony.co.jp/mawi/.
[49] DARPA 1999 intrusion detection evaluation data, http://www.ll.mit.edu/IST/ideval/docs/1999/schedule.html (last accessed March, 2008).
[50] Skype traces, http://tstat.tlc.polito.it/traces-skype.shtml (last accessed August, 2009).
[51] PacketShaper, http://www.packeteer.com/products/packetshaper/ (last accessed March, 2008).
[52] l7 filter, http://l7-filter.sourceforge.net/ (last accessed March, 2008).
[53] Wireshark, http://www.wireshark.org/ (Last accessed Sep, 2008).
[54] NetMate, http://www.ip-measurement.org/tools/netmate/.
[55] IETF, http://www3.ietf.org/proceedings/97apr/97apr-final/xrtftr70.htm.
[56] Libpcap, http://www.tcpdump.org/ (Last accessed Sep, 2008).
[57] WEKA software, http://www.cs.waikato.ac.nz/ml/weka/.
[58] A. McIntyre, M. Heywood, Cooperative problem decomposition in Pareto competitive classifier models of coevolution, in: European Conference on Genetic Programming, Vol. 4971 of Lecture Notes in Computer Science, 2008, pp. 289–300.
[59] R. Curry, Towards efficient training on large datasets for genetic programming, http://www.cs.dal.ca/~mheywood/Thesis/RCurry.pdf (2004).
[60] M. F. Brameier, W. Banzhaf, Linear Genetic Programming (Genetic and Evolutionary Computation), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.