J Netw Syst Manage (2015) 23:830–869
DOI 10.1007/s10922-014-9324-6

How Robust Can a Machine Learning Approach Be for Classifying Encrypted VoIP?

Riyad Alshammari · A. Nur Zincir-Heywood

Received: 6 May 2013 / Revised: 2 June 2014 / Accepted: 4 July 2014 /
Published online: 17 July 2014

Abstract The classification of encrypted network traffic represents an important issue for network management and security tasks including quality of service, firewall enforcement, and security. Traffic classification becomes more challenging since the traditional techniques, such as port numbers or Deep Packet Inspection, are ineffective against Peer-to-Peer Voice over Internet Protocol (VoIP) applications, which used non-standard ports and encryption. Moreover, traffic classification also represents a particularly challenging application domain for machine learning (ML). Solutions should ideally be both simple—therefore efficient to deploy—and accurate. Recent advances in ML provide the opportunity to decompose the original problem into a subset of classifiers with non-overlapping behaviors, in effect providing further insight into the problem domain and increasing the throughput of solutions. In this work, we investigate the robustness of an ML approach to classify encrypted traffic on not only different network traffic but also against evasion attacks. Our ML based approach only employs statistical network traffic flow features without using the Internet Protocol addresses, source/destination ports, and payload information to unveil encrypted VoIP applications in network traffic. What we mean by robust signatures is that the signatures learned by training on one network are still valid when they are applied to traffic coming from totally different locations, networks, time periods, and also against evasion attacks. The results on different network traces, as well as on the evasion of a Skype classifier, demonstrate that the performance of the signatures are very promising, which implies that the

R. Alshammari (✉)
College of Public Health and Health Informatics, King Saud Bin Abdulaziz University for Health Sciences, P.O. Box 22490, Riyadh 11426, Kingdom of Saudi Arabia
e-mail: alshammarir@ksau-hs.edu.sa

A. N. Zincir-Heywood
Faculty of Computer Science, Dalhousie University, Halifax, NS B3H 1W5, Canada
e-mail: zincir@cs.dal.ca

statistical information based on the network layer with the use of ML can achieve high classification accuracy and produce robust signatures.

Keywords Traffic identification · Security · Supervised learning · Evasion

1 Introduction

Traffic classification becomes a crucial requirement for network administrators to manage their network and help them to allocate expensive network bandwidth and resources to essential applications. Hence, network administrators are in need of efficient tools to manage, control and measure the traffic. However, managing network traffic requires huge resources to verify the traffic on the network, to ensure that organizational policies are met and to ensure security for the users. Identifying Internet Protocol (IP) network traffic according to the application type has the capability of resolving some of the complicated network management problems for organizations (enterprises) such as managing the bandwidth budget and ensuring quality of service objectives for critical applications.

Furthermore, traffic classification is 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 try to hide their behaviour either in encrypted or covert tunnels. For example, should some law enforcement agencies need to intercept or capture malicious traffic, e.g., child pornography traffic, from an Internet Service Provider (ISP), then the IP network traffic classification is a core part of the solution. Hypothetically, from a chief security officer's point of view, any application that hides its behaviour and avoids detection by encrypting its payload and implementing many methods to bypass firewalls or proxies is a risky application for sensitive information.

Thus, systems that can classify encrypted traffic represent a first step in identifying such malicious behaviours. Moreover, such systems can be useful as a forensic tool to identify applications used by malicious users whose data is collected/captured by law-enforcement units. In this case, establishing the classification of traffic types can reflect the current utilization of applications and services in a given traffic trace. In turn, this can help law-enforcement units to make a case for investigating the true intent of malicious users. Moreover, based on the traffic classification approach employed, user privacy can be maintained.

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 headers) while the second approach includes more sophisticated techniques based on Deep Packet Inspection (DPI) within the TCP or UDP payloads (visible payloads) 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 non-standard ports to bypass firewalls or circumvent operating system restrictions. New applications such as Skype have not registered port numbers with the 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 [2] showed that a classification based on the IANA port list is correct 70 % of the time. Madhukar and Williamson [3] confirmed that port number analysis misclassifies 30–70 % of their flow traffic.

On the other hand, the second approach, DPI, assumes the access to the payload of every packet. This technique can be extremely accurate when the payload is not encrypted. Sen et al. [4] demonstrate that classifying Peer-to-Peer (P2P) traffic based on payload signatures could reduce false positive and false negative rates by 5 %. Moreover, Moore and Papagiannaki [2] showed that using the entire payload can classify 100 % of packets correctly. However, DPI has many limitations. First, governments may regulate the use of payload and enforce constraints on its use since it can violate some organizational privacy policies or go against related privacy legislation. Secondly, examining the payload of a packet at the network speed is a computationally expensive task since the speed of networks, i.e., packet volumes transmitted through a network, is increasing all the time. Hence, deploying DPI, which works efficiently is challenging. Finally, the success of DPI is losing ground since new applications such as Skype or other Voice over Internet Protocol (VoIP) or P2P traffic that use techniques such as protocol encapsulation, payload encryption, and protocol obfuscation, imply that the payload is opaque. Thus, other techniques are required to increase the accuracy of network traffic classification.

As discussed earlier, since the traditional methods are ineffective for the new emerging P2P VoIP applications, many studies in the literature employ ML techniques using statistical flow information (features). Such features are usually derived from the information on the transport layer, which does not depend on port numbers or payload inspections. Moreover, they have shown promising results for classifying encrypted applications [5–8]. However, when ML techniques are deployed to extract signatures from traffic data, they require training data (where the traffic is labelled by application, i.e., ground truth) and a feature set, such as inter-arrival times or packet sizes, to represent the traffic. Therefore, ML classifiers are trained on the data sets to correlate feature values with the class label (application) to extract signatures (rules) that are then used to classify unknown traffic.

Recent research in this area focuses on the identification of efficient and effective classifiers. Different research groups have employed expert systems or various ML techniques such as Hidden Markov models, Naïve Bayesian models, AdaBoost, RIPPER, Decision Trees or Maximum Entropy methods to investigate this problem [2, 9–17]. Moreover, the limitations of port and payload based analysis have motivated the use of transport/flow layer statistics for traffic classification [5, 6, 8]. These techniques rely on the observation that different applications have distinct behaviour 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 between

P2P or encrypted applications accurately. Moreover, P2P and encrypted applications such as Skype use port numbers dynamically as well as using the same port number for multiple applications. This in return makes the problem even more challenging.

There are three commonly used ML techniques in the literature. These are supervised learning, unsupervised learning, and semi-supervised learning. Supervised learning is finding the correlation between the target class (labels) and the input feature to build a set of rules/models. Supervised learning requires labelled training data. Unsupervised learning is the clustering (grouping) of records that have similar characteristics according to the input features regardless of the classes (labels). Thus, unsupervised learning does not require labelled data. On the other hand, semi-supervised learning aims to understand how combining labelled and unlabelled data may change the learning behaviour, and how to design algorithms that take advantage of such a combination [18]. In this research, the focus is on the application of supervised learning techniques to an encrypted VoIP traffic classification, specifically classification of Skype traffic. The reason such an approach is taken is twofold: (1) automatically generating signatures (rules) is necessary to classify VoIP encrypted traffic, and (2) automating the process of selecting the most appropriate features /attributes for those signatures becomes possible by using supervised ML techniques. In this paper, we have employed three supervised ML algorithms: C5.0, AdaBoost, and Genetic Programming (GP). The reasons we have employed these ML algorithms include the following: previous work has reported good performance from these learning models in their respective studies [13, 14, 17]. In the form of rules, we observed these models to give the best solutions under different network conditions [10–12, 19]. 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 (features). Last, but not least, all three of these learning algorithms can generate solutions automatically in the form of rules that are easy to understand by human experts. We refer to these rules as the automatically generated signatures to identify the target application, e.g., Skype, in a given traffic log file. This is a very important property in order to employ the generated rules as signatures to classify traffic in practice. Furthermore, these learning models (C5.0, GP, and AdaBoost) provide human readable solutions, and hence, the solutions they generate are not a black box to the system administrators or network engineers. Additionally, other supervised learning algorithms (black box methods) such as Support Vector Machines (SVM) and Bayesian methods have significant memory overheads. Particularly, Bayesian methods require a lot of expertise to extract their potential. Conversely C5.0 addresses the memory overheads of C4.5 [20], making for a very robust implementation. Likewise, AdaBoost and GP manage the memory very well.

The use of ML techniques requires two major steps. Firstly, features need to be defined to describe the traffic data to the ML algorithms. In this case, features can be calculated over flows representing multiple packets. Secondly, the ML algorithms need to be trained to find patterns to correlate features to known traffic classes (supervised learning) and create models/rules to classify traffic. Every ML technique has a different schema to associate with a specific set of features for building the final model/solution. At the end of this phase, the best learner is selected based on the highest performance in terms of maximizing the Detection Rate (DR) and minimizing the False Detection Rate (FPR). Then, the best learner (solution) is employed to test the robustness/generalization of the rules/signatures learned on unseen data sets, which include different locations, different networks and different time periods. Callado et al. [21] showed experimentally that the performance of ML algorithms vary on different network traces since each network has different behaviour and characteristics. Therefore, it is important to have different training and testing network traffic data for evaluating the performance of the ML algorithms in particularly their robustness.

In summary, the primary objective of this research is to explore the robustness of automatically generated signatures for encrypted VoIP traffic. In this case, robustness analysis consists of unseen data sets including (1) different locations (networks), (2) different time periods, and (3) evasion attacks.

The rest of this paper is organized as follows. Related work is discussed in Sect. 2. Section 3 introduces the machine learning algorithms, data sets, the feature set, and the evaluation method employed. Section 4 presents the experimental results. Finally, conclusions are drawn and future work is discussed in Sect. 5.

2 Literature Survey

Traffic classification has been interested to the research community for years. There are many works that described methods, techniques, and tools to classify network traffic. Two survey papers summarize research in the IP traffic classification from the period of 1997–2008. First, Nguyen and Armitage [22] focus on research in the literature that use ML techniques to classify IP network traffic. Secondly, Callado et al. [23] presented the challenges in the area of IP traffic analysis and application classification. They divided the techniques for traffic analysis and classification into two categories: packet based and flow based.

In the literature, Bonfiglio et al. [24] present one of the earlier studies in classifying Skype traffic using supervised learning techniques. They introduced two approaches for classifying Skype traffic. The first approach is to classify Skype traffic based on Pearson's Chi-Square (χ^2) 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 traffic based on the Naïve Bayesian Classifier using the packet arrival rate and packet length. They obtained the best results when the first and second approaches were combined. They achieved approximately a 1 % false positive rate and a 2–29 % false negative rate, depending on the data sets. However, they employed these under a payload-based classification scheme and used a priori knowledge for UDP detection. Freire et al. [25] studied detecting Skype flows in Web traffic by using metrics derived from the χ^2 value and the Kolmogorov–Smirnov distance using features based on Web request/response sizes, the number of requests and the time taken to detect

Skype and Gtalk traffic. They achieved a 100 % DR and a 5 % FPR for both applications. Recently, Este et al. [26] applied SVM for classifying only TCP bi-directional flows relying mainly on the packet size as the main feature. The SVM models are able to classify multiple applications such as HTTP, HTTPS, BitTorrent, e-Donkey, Kazaa, Gnutella and, MSN. The classification method is based on two phases. For the first phase, they build a one-class SVM classifier in which the classifier can determine the application of the TCP flow based on where the flow feature values fall in SVM hyperplanes (surfaces). For the second phase, they built a multi-class SVM based classifier. The second phase is called when the TCP flow falls on more than one hyperplane in order to determine the correct application class. Furthermore, if the flow falls outside the hyperplane, the flow is marked as 'unknown'. They tested their methods on three network traces, captured at different locations. They were able to achieve high performance on the e-Donkey flows but had poor results on other P2P applications such as Kazza and Gnutella. Moreover, they did not study the robustness of the SVM models.

Unsupervised learning methods have been used in network traffic classification as well. Bernaille et al. [8] used an unsupervised learning method to cluster the network traffic in order to label it according to the application protocols. They clustered the first five packets of TCP flows based on the packet size in each connection. They used the *K*-Means algorithm with the Euclidean distance to build an online classifier consisting of 50 clusters to classify only TCP network flows. The classifier begins by building the flows based on the 5-tuple (Protocol, Source/Destination IP addresses and Source/Destination port numbers) from the TCP header and calculating packet sizes. Then, the classifier searches all 50 clusters to label the coming traffic (new flow) according to the application type. They were able to accurately classify more than 80 % of the P2P application traffic. In particular, they achieved 95 % accuracy for Kazaa and 84 % accuracy for e-Donkey traffic. However, the classifier has problems handling similar flow sizes employed by different applications, basically labelling the flows the same way.

Erman et al. [27] apply a semi-supervised technique for classifying network flow traffic as the Web, FTP, and P2P file sharing. The semi-supervised learning method consists of two steps. During the first step, the *K*-Means algorithm with Euclidean distance is used to cluster traffic. The clusters contain pre-labelled flows and unlabelled flows. The second step involves using the maximum likelihood estimate for the pre-labelled flows within a cluster to map the cluster into a known traffic application. Clusters that have no pre-labelled flows are mapped into the 'unknown class'. They applied the backward greedy feature selection algorithm to choose eleven flow features. They achieved a best performance of $\approx$98 % flow classification accuracy and $\approx$93 % byte accuracy. Their performance metric is based only on accuracy, which is the number of correctly classified instances divided by the total number of instances, rather than providing classification results based on the false positive rate as well. Unfortunately, this may be misleading, especially on unbalanced data sets in which the data set consists of, say, two classes only (in a total of one hundred instances), 10 instances of FTP and 90 instances of P2P. By labelling everything as the major class, a classifier can achieve 90 % accuracy but the false positive rate for P2P would be 100 %. Bacquet et al. [28] employed five unsupervised learning algorithms that are Basic K-Means, Semi-supervised K-Means, DBSCAN, EM, and MOGA for detecting encrypted traffic, namely, SSH. Results showed that MOGA performed better than other unsupervised learning algorithms. MOGA achieved 93.5 % DR and 0.7 % FPR.

Recently, Iliofotou et al. [29] employed Traffic Dispersion Graphs (TDGs) for classifying P2P traffic, e.g., Gnutella, e-Donkey and BitTorrent. Their approach worked by grouping the first sixteen bytes of the payload using the *K*-Means algorithm. These bytes act as categorizing features ranging from 0 to 255. Then, the TDGs are used to classify the clusters. They applied their approach on backbone traffic collected from different sites and showed that they were able to classify 90 % of P2P traffic with an average precision of 95 %.

In terms of the generalization/robustness of the classifier, Park et al. [30] pointed out the importance of finding a robust classifier. They used different data for training and testing for network traffic classification that were different in terms of time periods and locations. Their scheme consists of using Genetic Algorithms (GA) for feature set reduction and a decision tree as a classifier. However, they did not provide the results of individual applications, but rather, they provided the overall results. Hu et al. [31] build a behavioural profiling 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 and statistical flow features. They used an 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 has different than the training data set of $\approx$98 and $\approx$97 % for PPlive over TCP and UDP, respectively; and $\approx$94 and $\approx$96 % for BitTorrent over TCP and UDP; respectively. However, they do not report true positive and false positive rates and discuss in theory what might happen against evasion attacks. On the other hand, Wright et al. [32] used traffic morphing techniques to make one application traffic look similar to another application traffic by padding the payload of a packet. Their results showed the morphing techniques were able to reduce the accuracy for the VoIP language classifier, designed by Wright et al. [33], from 71 to 30 % and to reduce a web page classifier, designed by Liberatore and Levine [34] from 98 to 5 %. However, the results of the classifier enhanced significantly when the morphed data were included in the training data set and the experiments were repeated.

Thus in this research, we aim to perform an investigation of C5.0, GP, and AdaBoost based classifiers on the identification of VoIP, e.g., Skype, encrypted traffic as well as explore their robustness. We focus on the robustness by not only evaluating the classifiers on (1) unseen data from the same network but by also evaluating them against unseen data captured from different networks/locations at (2) different time periods as well as against (3) potential evasion (bypassing classifiers) attacks. Even though in some of the previous work, evaluations against different network traffic traces were reported [30] or discussions against evasion attacks are presented [31, 33], to the best of our knowledge, this work is the first time where generalization/robustness of classifiers are evaluated under the three aforementioned conditions to specifically classify Skype P2P VoIP traffic.

2.1 Overview of VoIP Applications

In this work, we are interested in identifying VoIP Skype by using features extracted from the Transport layer (layer 4 of the Internet protocol stack) and the Network layer (layer 3 of the Internet protocol stack). Skype [35] 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 provides similar functions as MSN and Yahoo instant message applications, the fundamental protocol and techniques it operates are completely different. Since the Skype protocol is proprietary and an extensive use of cryptography is implemented by the Skype creators, understanding Skype protocol is a difficult task. Moreover, Skype employs a number of methods to evade network address translations (NAT) and firewall restrictions that increase the difficulty of understanding the Skype protocol even more.

Skype is based on a hybrid P2P/C-S architecture except a user's authentication is performed based on a central architecture (client–server model via public key mechanisms). After authentication is completed, most communication is performed on the P2P network. Therefore, user information and search queries are 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 the TCP or UDP at the transport layer to provide services to users such as voice and video calls, file transfers, chats, and conference calls. The communication among peers (users) on the P2P network is established via an IP paradigm. A more detailed description of Skype protocol, behaviour, internals, analysis, and the quality of video and voice calls on different scenarios can be found in [36–39].

3 Methodology

In this research, the focus is on the application of supervised ML based techniques to network traffic classification, specifically classification of Skype encrypted VoIP traffic. To this end, three different supervised machine learning algorithms, AdaBoost, C5.0, and GP, are evaluated to automatically generate signatures to identify VoIP traffic robustly. The ML techniques require a number of steps such as selection of features, labelling of the data set, training of the learning algorithms, and testing the solutions produced by the ML algorithms that are discussed in detail in this section.

3.1 Flow-Based Feature Set

In this work, we represent the network traffic as a bidirectional flow connection between two hosts where the two hosts have the same 5-tuple (source and destination port numbers, source and destination IP addresses and the protocol). In these flows, the client-to-server connections represent the forward direction while the server-to-client connections represent the backward direction. Moreover, the flow time-out value employed in this work is 600 s as specified by the Internet Engineering Task Force (IETF) [40]. TCP flows are ended either by flow time out or by connection teardown while UDP flows are ended by flow time out. We used the NetMate tool set [41, 42] to generate the flows and calculated the statistical feature values, Table 1. Furthermore, we only include flows that have at least one packet in both directions and have a payload of at least one byte. In this work we are interested in identifying encrypted VoIP applications at the Application layer by using information (feature sets) extracted from the Transport layer, e.g., time to calculate inter-arrival time and the Network layer, e.g., protocol.

Table 1 Flow feature employed

	Abbreviation	Feature name
1	min_fiat	Minimum of forward inter-arrival time
2	mean_fiat	Mean of forward inter-arrival time
3	max_fiat	Maximum of forward inter-arrival time
4	std_fiat	Standard deviation of forward inter-arrival times
5	min_biat	Minimum of backward inter-arrival time
6	mean_biat	Mean backward inter-arrival time
7	max_biat	Maximum of backward inter-arrival time
8	std_biat	Standard deviation of backward inter-arrival times
9	min_fpkt	Minimum of forward packet length
10	mean_fpkt	Mean of forward packet length
11	max_fpkt	Maximum of forward packet length
12	std_fpkt	Standard deviation of forward packet length
13	min_bpkt	Minimum of backward packet length
14	mean_bpkt	Mean of backward packet length
15	max_bpkt	Maximum of backward packet length
16	std_bpkt	Standard deviation of backward packet length
17	proto	Protocol
18	Duration	Total duration
19	f_packets	Number of packets in forward direction
20	f_bytes	Number of bytes in forward direction
21	b_packts	Number of packets in backward direction
22	b_bytes	Number of bytes in backward direction

3.2 Machine Learning Algorithms Deployed

These are three learning algorithms employed in this paper. These are C5.0, AdaBoost, and GP. The C5.0 [43] is the commercial decision tree algorithm developed from the famous C4.5 decision tree algorithm. C5.0 includes all the properties of C4.5 and has additional new technologies such as boosting. The major advantage of C5.0 over C4.5 is efficiency, otherwise both algorithms remain the same [20]. C5.0 builds its solution by recursively splinting the input space into regions where the splits consider to be pure for all branches. The C5.0 uses entropy to calculate the proportion of exemplars corresponding to the number of classes in the training data. In the case of an impure split, the exemplars are separated to decrease the impurity. The next stage is calculating the information gain for each feature to reduce the entropy in the training data. Additional information on this algorithm can be found in [44].

On the other hand, the AdaBoost algorithm is a meta-learning algorithm that builds its solution incrementally by boosting weak learning classes to strong leaning classes from the training data set.The classes are built by the intersection of many weak simple classes (decision stumps) by using a voting scheme. The AdaBoost algorithm generates many hypotheses where each decision stump would return $+1$ or -1. Additional information on this algorithm can be found in [44].

Finally, the Symbiotic Bid-Based (SBB) GP technique, which is part of the team based GP family, is also employed in this work. SBB depends heavily on coevolution [45] to build its model by employing three different populations, namely teams, points, and learners. A symbiotic relation exists between the learner population and the team population where a bidding strategy exists between the learner population and the team population. A linear representation is employed as a bidding strategy of the learner population of individuals. Furthermore, the individuals in the team populations bid against each other to compete on the training data. The point population is responsible for the competitive coevolutionary relationship between the team population and the point population that can scale the evolution on big data sets [46]. Additional information on the SBB based GP algorithm can be found in [45].

3.3 Evaluation of Machine Learning Algorithms

Two evaluation criteria were used in traffic classification to measure the performance of the learning algorithms. These are the DR and the False Positive Rate (FPR). The DR, Eq. 1, reflects the total number of flows that are correctly classified from the in-class (the ones that the algorithm aims to classify)

$$DR = \frac{TP}{TP + FN} \tag{1}$$

whereas the FPR, Eq. 2, reflects the total number of out-class flows that are classified incorrectly as in-class.

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

The desirable outcomes were to obtain a high percentage value for the DR and a low percentage value for the FPR. Moreover, the False Negative (FN) rate represents the total number of in-class flows that are classified as out-class flows.

In this paper, 50 runs were used to train each of the learning algorithms on each training data set to generate different models. To this end, we used 50 different confidence factors for C5.0, 50 different weight thresholds for AdaBoost, and 50 different population initializations for SBB-GP. Weka [47] was used for running the AdaBoost, the Linux model given at [43] was used for running the C5.0, and the C++ implementation given at [48] was used for running the SBB-GP learning algorithms. The parameters of the three algorithms are listed in Tables 2, 3, and 4, respectively. We used 50 runs for each of the algorithm on each training data set to ensure that the outcomes were not based on one off trails but rather were based on statistically significant trials. Furthermore, the non-dominated solutions were selected out of the 50 models. The non-dominated solutions are the distinctive solutions that ranked the best model based on the high value of DR and the low value of FPR out of all models. Then, the best learner (one model only) out of the non-dominated learners was chosen manually based on the highest performance (again the highest DR and the lowest FPR).

3.4 Robustness of Machine Learning Algorithms

In most cases, researchers have evaluated the performance of their approaches on traces from the same network on which the model was trained; albeit, the testing data sets were

Table 2 C5.0 parameterization

	Description	Value
r	Use rule-based classifiers	True
b	Use boosting	False
p	Use soft thresholds	True
e	Focus on errors	True
s	Find subset tests for discrete attributes	False
c	Confidence factor for pruning	5–54

Table 3 Weka parameterization for AdaBoost

	Description	Value
classifier	The base classifier to be used	DecisionStump
numIterations	Number of iterations	10
seed	The random seed number	1
useResampling	Use resampling instead of reweighting	False
weightThreshold	Weight Threshold (default 100)	10–250

Table 4 SBB based GP parameterization

	Description	Value
P_{size}	Point population size	90
M_{size}	Team population size	90
t_{max}	Number of generations	30,000
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

unseen during training. On the other hand, we believe that it is important to evaluate the robustness/generalization of the solutions in traffic classification, e.g, VoIP. In this research, the robustness/generalization is defined from three perspectives. These are testing on (1) unseen data from different locations and network infrastructures; (2) unseen data from different time periods; and (3) unseen altered data by padding/morphing, i.e., evasion attacks. To the best of our knowledge, this is the first research to evaluate robustness of signatures produced by ML algorithms on all three criteria.

3.5 Traces Deployed

To show the effectiveness of the proposed approach, completely different datasets are employed for training and testing the classifiers. Moreover, solution robustness is assessed by training on a data set from one location (hereafter denoted Univ2007 training trace) but by testing on data sets from other locations (University 2007 Test partition, University 2010, ITALY, NIMS2, NIMS3, and IPv6 traces, which were captured between 2000 and 2010).

3.5.1 University Traces

Two university traces were captured on the Dalhousie University Campus network by the Information Technology Services Centre (ITS) in January 2007 and May 2010 (Univ2007 and Univ2010). Dalhousie is one of the biggest universities in the Atlantic region of Canada. The ITS is responsible for all the networking on the campus. The Dalhousie network is connected to the Internet via a full-duplex T1 fiber link. Given privacy issues, data is filtered to scramble the IP addresses and each packet is truncated further to the end of the IP header so that all payload was excluded. Moreover, the checksums are set to zero since they could leak information from the short packets. However, any information regarding the size of the packet is left intact. Finally, both the Univ2007 and Univ2010 traces are labelled by ITS using a commercial classification tool called PacketShaper, which is a deep packet analyzer [49].

3.5.2 ITALY Traces

The ITALY data set consists of 96 h of Skype Traffic over the TCP and UDP protocols [50]. The data set was captured on the main link at the Politecnico di Torino University campus. TCP Statistic and Analysis Tool (Tstat) and the traffic classification method employed in [24] were used to label the traffic. As described in Sect. 2, the creators of this data set classified Skype traffic based on a DPI and per-host analysis. In this research, all the Skype traces, which were captured in the Politecnico di Torino main link were employed. These were (1) End-to-End (E2E) voice only and voice-video class (Skype UDPE2E); (2) Skype out calls (Skype UDPE2O); (3) Signaling connections only (Skype UDPSIG); and (4) End-to-End and Skype out calls (Skype TCPE2X). The first trace, which was captured over UDP, consists of voice only calls as well as voice plus video calls. The fourth trace was captured over TCP and consists of voice only calls.

3.5.3 NIMS2 Traces

VoIP traffic was generated using different applications on a testbed set up in the NIMS Lab in 2009 in Computer Science at Dalhousie University. This testbed involved several PCs connected through the Internet and several network scenarios were emulated using many popular VoIP applications, e.g., Gtalk, Primus, Yahoo messenger. The focus was on Gtalk traffic and how Gtalk reacts to different network restrictions was observed. Moreover, the effects (if any) of different types of access technologies, i.e., WiFi versus Ethernet, were investigated as well as their different combinations. In 2009, over two hundred experiments were conducted, equivalent to more than 50 h of VoIP traffic and non-VoIP traffic. This data set was made public at [51].

For this work, a Gtalk client was installed on each of the three Windows XP machines. The first machine was a Pentium 4 2.4 GHz Core 2 Duo with 2 GB RAM; the second machine was a Pentium 4 2 MHz Core 2 Duo with 2 GB RAM, and the third machine was a MacBook 2 GHz Intel Core 2 Duo with 2 GB RAM. Two machines had a 10/100 Mb/s Ethernet connection and the third machine had a wireless 10/100 Mb/s card. Furthermore, one was connected to a 1 GB/s network while the others were connected to a 10/100 Mb/s network. All three machines had Windows XP Service Packet 2 and all experiments were done using the Gtalk client version 1.0.0.104. In all experiments, Gtalk traffic was captured from both ends. In all cases, the experiments were performed under several different network scenarios (Fig. 1). These scenarios include (1) Firewall restrictions at one user end and no restrictions at the other end; (2) Firewall restrictions at both ends; (3) No restrictions at both ends; (4) Use of wireless and wire-line connections; (5) Blocking of all UDP connections, and (6) Blocking of all TCP connections. It should be noted here that during these experiments all the Internet communications went through the network's firewall. The firewall was configured to either block or to permit access to the following restrictions: (1) block everything, or (2) permit limited well-known port numbers: 22, 53, 80, and 443. Wireshark [52] and NetPeeker [53] were used to monitor and control network traffic. NetPeeker was used to block ports and to allow

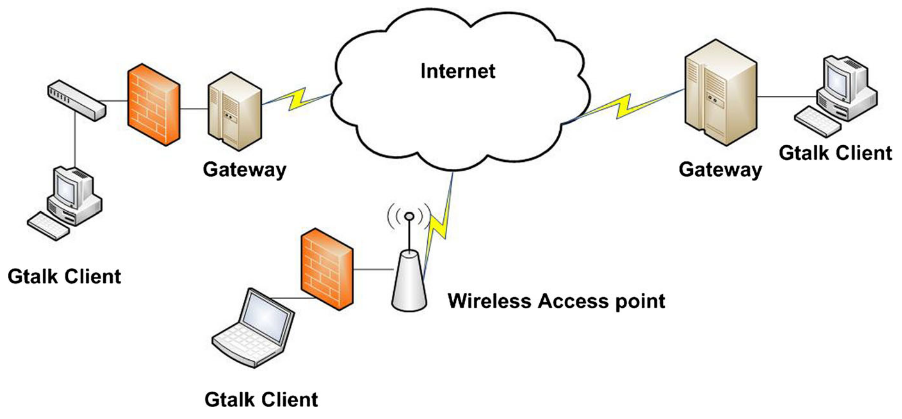


Fig. 1 Network setup with restrictions

either both TCP and UDP traffic, or only UDP or TCP traffic in order to analyze the behaviour of the Gtalk client. On the other hand, Wireshark was used to capture traffic from both ends of the communication.

The general call setup between the caller and callee for voice calls was as follows: caller transmited a standard audio file to callee. An English spoken text (male and female audio files) without noise and a sample rate of 8 Hz was used, which was encoded with 16 bits per sample and which could be download at [54]. The wav-file was played and then the output of the Windows media player was used as input for Gtalk, Primus [soft Talk Broadband (softTBB)] and Yahoo messenger (Encrypted with Zfone in 2009) with clients using a microphone. Wireshark was used to capture the traffic from both users' ends. We are different than [38, 39] for generating traffic in terms of: (1) we did not use an emulator; (2) our traffic was generated on the Internet; (3) we monitored and captured traffic at both ends of the communication; and (4) our scenarios were different [we focused more on firewall restrictions, protocol preferences (UPD or TCP)], and blocking ports while the two papers focused on congesting control, jitter and packet loss, etc. However, we had similarities in some cases such as connecting a microphone to play the audio file.

Furthermore, Yahoo messenger traffic was generated as well (encrypted with Zfone in 2009 only). Zfone traffic is another encrypted VoIP traffic that was generated. Zfone [55] is software that secures VoIP calls over the Internet. Zfone works by intercepting all the unencrypted VoIP channels and protecting the VoIP channels securely by encrypting all the VoIP packets. Zfone is the user interface of the Zimmermann Real-time Transport Protocol (ZRTP) [56]. ZRTP uses Diffie-Hellman to exchange keys over the Real-time Transport Protocol (RTP) packet stream (creating secure RTP sessions). It encrypts the payload of a packet using standard cryptographic algorithms such as the Advanced Encryption Standard (AES) or Rivest Shamir Adleman (RSA) algorithms. Zfone was used to secure all Yahoo Messenger audio calls. Zfone detects Yahoo packets and encrypts them as they are sent by the caller machine and detects the encrypted packets received by the callee machine and decrypts them.

Non-encrypted VoIP traffic was generated using a Primus Session Initiation Protocol (SIP) client [57]. Primus softTBB was used to make calls to a Public Switched Telephone Network (PSTN) for voice services (hard line phone) and mobil cell phone. The softTBB client runs on a PC or a laptop and connects to the Primus SIP Network over the Internet. Depending on what is called, i.e., a mobile phone or a PSTN phone, the Primus SIP network routes the calls to the final destination differently.

A call to a PSTN phone is routed to the nearby Primus Voice gateway that converts the calls between the VoIP network and the PSTN network. For a call to a cell phone that is subscribed to the Bell General Packet Radio Service (GPRS) and Universal Mobile Telecommunication System (UMTS) network, the route is more complex. According to the GPRS/UMTS specification [58], the main components of the GPRS/UMTS network are base stations and gateways connected to the Internet. In this case, first, the cell phone registers with the base station. Then, the base station is connected to the Serving Gateway Support Node (SGSN), which is connected to the Gateway GPRS Support Node (GGSN) inside the Bell GPRA/UMTS network. Finally, the GGSN is the first node responsible for processing IP packets from the Internet to the mobile network and vice versa. To establish the call between a Primus softTBB and a Bell mobile phone, the call is routed through the Primus SIP network through the Internet to the Bell GGSN gateway. In all cases, it was possible to listen to the call on the PSTN phone and the mobile phone. All communications were done without encryption and the traffic was captured using Wireshark only at the machine where softTBB was running, since permission to capture traffic with Primus or Bell companies was not an option. In this case, it was a deliberate choice not to encrypt the traffic in order to have different mixtures of VoIP traffic in the traces, i.e., both encrypted Gtalk, Skype, and Yahoo with Zfone and non-encrypted Primus.

3.5.4 NIMS3 Traces

Further VoIP traffic was generated in 2010 using the same testbed used for the NIMS2 traces. For the NIMS3 traces Gtalk and Primus traffic was generated using the same setup as in the NIMS2 traces (Sect. 3.5.3) and other background traffic, such as torrent traffic and web TV and radio, were included. However in 2010, Yahoo began to use encryption to secure its traffic. Therefore Zfone was not needed for encrypting the Yahoo traffic since it can be used only with a non-encrypted payload. Online web radio media stream (non-encrypted) traffic was captured as well. Also, the same methodology was used to capture a TV media stream channel broadcast on the web. Moreover, Torrent traffic was captured by installing a Torrent client to download a free Linux operating system distribution. Several hours of traffic were captured for each application.

Furthermore, due to wide usage of the Virtual Private Network (VPN) technology by companies and users who want to secure their communications, VPN traffic was generated as well and included in these NIMS3 traces. VPN technology can tunnel any traffic and quarantine the privacy and security between the two end points. Two VPN technologies were applied regularly for establishing VPN tunnels: Layer 2

Tunneling Protocol/Internet Protocol Security (L2TP/IPSec) and Secure Socket Layer Transport Layer Security (SSL/TLS). Both technologies provide encrypted and secure communications with different implementations. L2TP enables the encapsulation of Ethernet frames into UDP packets. To ensure the privacy of the packets, L2TP is combined with IPSec. IPSec comes with two configurations: the (1) Encapsulated Security Payload (ESP) and the (2) Authentication Header (AH). The ESP protects the payload by using encryption algorithms, e.g., Blowfish, while the AH protects the IP packet header by computing a cryptographic checksum and hashing. The ESP was chosen for two reasons: (1) authentication does not quarantine encryption of the payload; and (2) the VoIP applications employed in this thesis encrypt their payload. The second common technology for setting up a VPN tunnel is SSL/TLS, which is used typically for securing HTTP connections but which can be used as well to create a tunnel for transfering any traffic between two machines. The Mac OS X implementation of L2TP/IPSec was used to set a connection between two machines for the first VPN technology and a commercial tool called VPN-X [59], which employs the second VPN technology was used as well. For both VPN technologies a VPN tunnel was set up between two machines and their traffic was captured. Using the two VPN setups it was possible to view and share files, to browse web pages and to send chat messages. This testbed traffic has been made publicly available to the research community at [51], too.

3.5.5 IPv6 Traces

IPv6 is a version of IP and is intended to succeed IPV4 since the IPv4 address space is running out of addresses. The IPv6 address space is 128 bits long and provides privacy and security for communication on the network by applying encryption. IPv6 allows varieties of encryption algorithms. Kent and Atkinson define the IPv6 architecture [60]. Since IPv6 encrypts the packet and has a large address space, which will not be running out in the next few years, IPv6 traces are included as test data sets to evaluate the robustness of the signatures. Public IPv6 traces captured by the Measurement and Analysis on the WIDE Internet working group (MAWI) [61] in 2000 and 2009 were used. The IPv6 traces were captured daily at an IPv6 cable connected to the 6Bone testbed. The 6Bone [62] testbed was established by the IETF in 1996 to test IPv6 and ease the transition of the Internet to IPv6. The IPv6 traced a total of more than 9 h and include a several applications running over TCP and UDP protocols such as HTTP, HTTPS, SMTP, POP3, SSH, and DNS as well as traffic from other protocols, e.g., ICMP.

3.5.6 Statistics of Traces

Brief statistics on the traffic data collected are given in Table 5. This shows that all the data sets have different general properties in terms of the number of packets and flows and the size of the traces.

Table 5 Overview (in millions) of network traces employed

	Packets	Bytes	Flows
Univ2007	336M	212,931M	28M
Univ2010	1,857M	1,347,191M	46M
Skype UDPE2E	39M	7,840M	295,811
Skype UDPE2O	3M	188M	2,601
Skype UDPSIG	71M	15,141M	50M
Skype TCPE2X	2M	302M	13,306
NIMS2: GTALK2009	34M	6,485M	190,665
NIMS2: PRIMUS2009	2M	384M	7,529
NIMS2: Zfone2009	1M	138M	28,553
NIMS3: GTALK2010	384M	1,256M	14,847
NIMS3: PRIMUS2010	7M	1,367M	21,802
NIMS3: YAHOO2010	8M	1,080M	23,239
NIMS3: Torrent2010	21M	17,791M	412,345
NIMS3: Radio2010 stream	332,183	272M	2,236
NIMS3: TV2010 stream	5M	4,941M	1,803
NIMS3: VPN2010	32,079M	26,728M	74,302
IPv6	808M	825M	1,151

4 Results of Experiments

In this section, to show the effectiveness of our proposed approach in finding robust signatures, we use completely different datasets for training and testing the classifiers in order to provide some measure of classifier generalization (robustness). Thus, solution robustness is assessed by training on a data set from one location (Univ2007 training trace) but by testing on data sets from different locations, different networks, and different time periods (Univ2007 Test partition, Univ2010, NIMS2, NIMS3, ITALY, and IPv6 which were captured in 2007, 2010, 2009, 2010, 2006, and in both 2000 and 2009 for IPv6, respectively). The reasons for sampling the training data set only from Univ2007 traces was threefold: (1) We wanted to evaluate the ML classifiers for the VoIP traffic in terms of how accurately they might work in the future, so we trained on the older traces and tested on the newer traces; (2) We wanted to find out if the Skype P2P VoIP traffic changed its behavoir (extracted signatures) from time to time; and (3) We wanted to find out the effect of time, location, and different network architecture on the robustness of the classifier (extracted signatures). Issues of data representation were addressed by employing flow based features only without using IP addresses, port numbers, or payload data.

4.1 Results of Flow Experiments for Skype Identification: Robustness of Signatures

The training data set was generated by sampling randomly with a uniform probability of 6 % from both classes (in-class and out-class). Moreover, since there were no Skype TCP flows in the Univ2007, we sampled 6 % from the ITALY (Skype_TCP_E2X) traffic. In total, the training data set consisted of 1,739,588 flows.

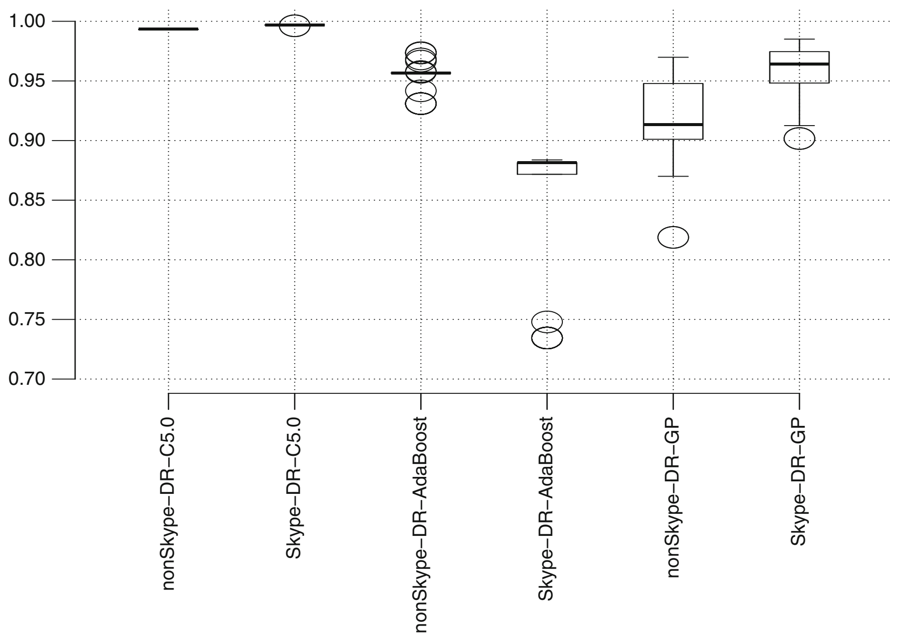
Figure 2 summarizes solutions for the three ML algorithms on the training trace: all model construction took place on the Univ2007 training partition. Testing evaluation was conducted under the Univ2007 Test partition, Univ2010, ITALY, NIMS2, NIMS3 and IPv6 traces where none were encountered during training. Naturally, the Univ2007 Test partition reflected the training behavior more closely than the other test network traces.

Results presented in Fig. 2 and the one-way ANOVA statistical analysis test in Tables 6 and 7 illustrate that the C5.0-based classification approach was much better than other algorithms employed in identifying the Skype flow traffic based on the training data set. The boxplot demonstrates the diversity of performance (in terms of DR and FPR) for all 50 models on the training trace for each ML algorithm (Fig. 2). On average, C5.0 was much better than other ML algorithms in terms of a high DR and low FPR.

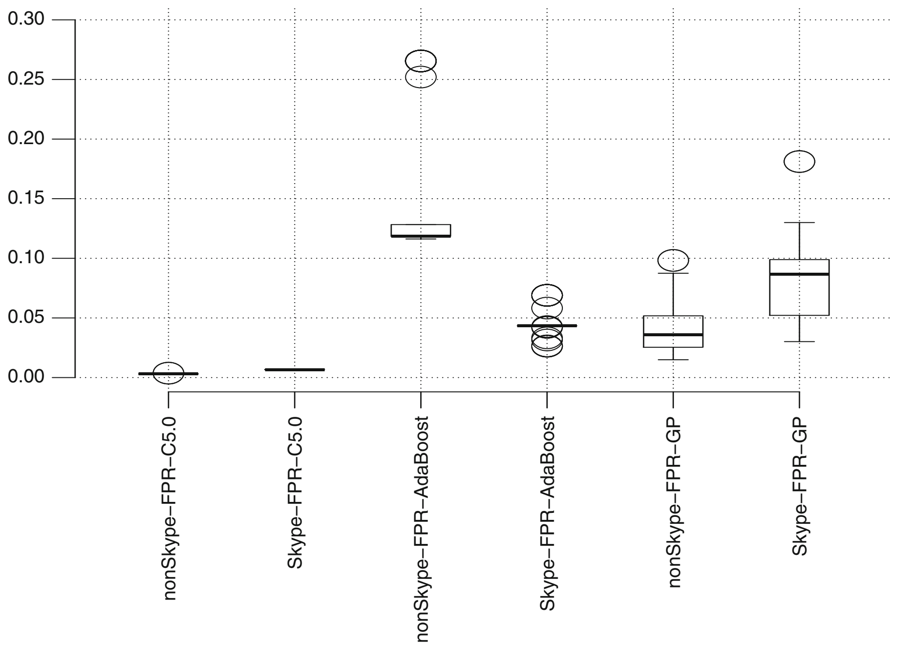
To select the best trained model for each ML algorithm the training performance was plotted using the scatter plot for each of the three ML algorithms. Figures 3, 4 and 5 summarize the solutions. For Skype, there are four solutions which are non-dominated for GP, four which are non-dominated for AdaBoost and seven which are non-dominated for C5.0. In Fig. 4, there is a jump and dip from 0.03 to 0.07 % FPR. This is due to changing of the weight threshold from 10 to 250, which would effect on the model generated by AdaBoost, and hence, the effect on the FPR rate. The best performing solution in terms of high DR and low FPR was selected out of these non-dominated solutions for GP, AdaBoost and C5.0 and then evaluated on the test data sets.

Tables 8, 9, 10, 11, and 12 list the results for the three ML algorithms on the training and independent test traces. For Skype flows, results show that the C5.0-based classifier performed better than the other classifiers on the Univ2007 Test partition while the GP-based classifier performed better than all the others on the Univ2010 test traces. The C5.0-based classifier achieved ≈100 % DR and ≈1 % FPR on the Univ2007 Test partition and ≈80 % DR and ≈6 % FPR on the Univ2010 traces. By introducing the entirely independent test sets in terms of location, time, and network infrastructure (the Univ2010, ITALY, NIMS2 & 3 and IPv6 traces), the performance of the classifiers was evaluated from a different perspective. In this case, as the results in Table 9 show, AdaBoost overlearned the properties implicit in the training partition since the performance of the AdaBoost-based classifier dropped to a 0 % DR on the ITALY traces. Moreover, C5.0 and GP based classifiers were observed to provide good performances on both the Univ2010 and ITALY traces, whereas the AdaBoost and the C5.0 based classifiers achieved very good performances on the NIMS2, NIMS3 and IPv6 test data sets.

The C5.0-based classifier achieved ≈99 % DR on the ITALY-TCPE2X traces, ≈89 % DR on the ITALY-UDPE2E traces, ≈53 % DR on the ITALY-UDPE2O traces and ≈92 % on the ITALY-UDPSIG whereas GP achieved ≈71 % DR on the ITALY-TCPE2X traces, ≈61 % DR on the ITALY-UDPE2E traces, ≈94 % DR on



(a)



(b)

Fig. 2 Results for Skype flow traffic on the training data sets. **a** DR, **b**FPR

Table 6 A one-way ANOVA statistical analysis test for the mean DR for the three ML algorithms using the flow-based feature set on training trace

Source	SS	*df*	MS	F	Prob >F
Columns	0.6150	2	0.3075	209.2938	9.7519e−44
Error	0.2160	147	0.0015		
Total	0.8310	149			

SS sum squares, *df* degree of freedom, *MS* mean squares, *F* F-statistic

Table 7 A one-way ANOVA statistical analysis test for the mean FPR for the three ML algorithms using the flow-based feature set on training trace

Source	SS	*df*	MS	F	Prob >F
Columns	0.1433	2	0.0717	182.2266	1.5870e−40
Error	0.0578	147	3.9333e−04		
Total	0.2012	149			

SS sum squares, *df* degree of freedom, *MS* mean squares, *F* F-statistic

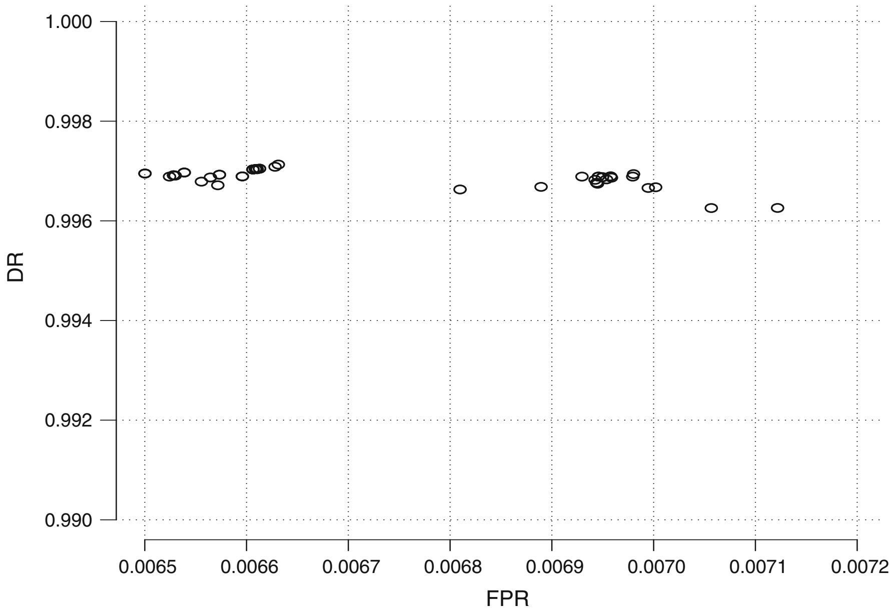


Fig. 3 Scatter plot for the C5.0 classifier for training performance using the flow-based feature set for Skype detection (DR versus FPR)

the ITALY-UDPE2O traces and ≈61 % on the ITALY-UDPSIG (Table 9). However, AdaBoost performed better than C5.0 and GP on the NIMS test traces and the IPv6 traces (Tables 10, 11, 12).

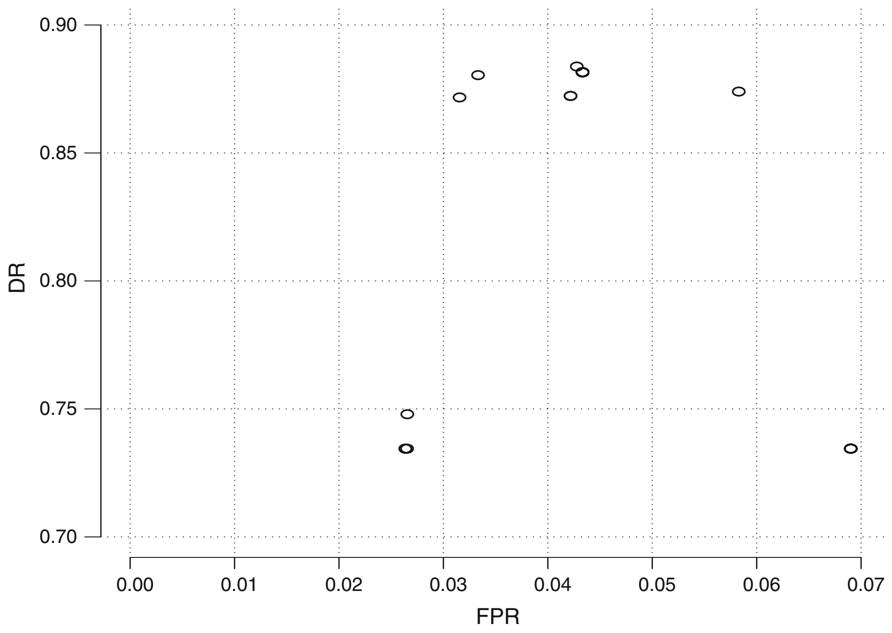


Fig. 4 Scatter Plot for the AdaBoost Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR)

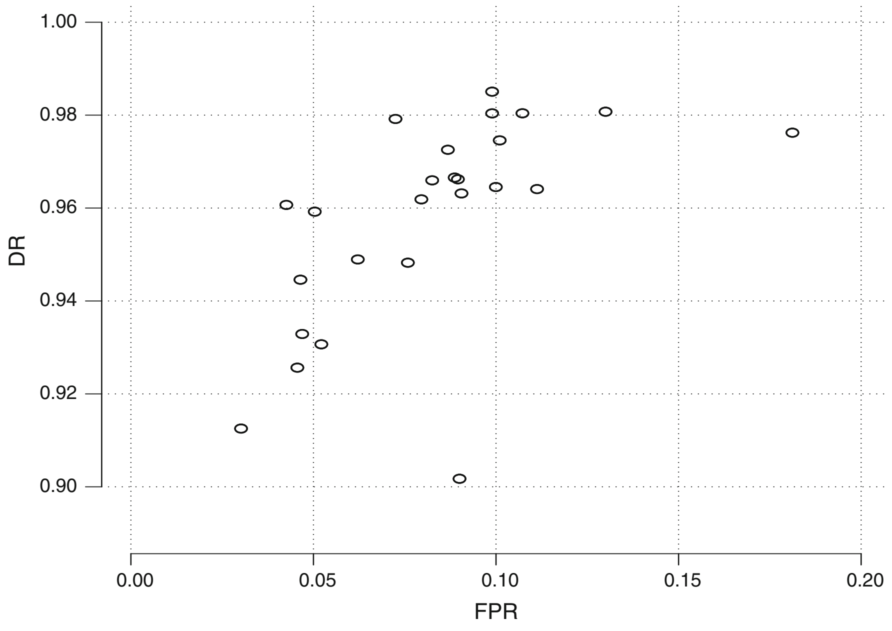


Fig. 5 Scatter plot for the GP classifier for training performance using the flow-based feature set for Skype detection (DR versus FPR)

Table 8 Best model results on training data set for the flow-based feature set for Skype detection (training and testing on Dalhousie University data sets)

	C5.0		AdaBoost		GP	
	DR	FPR	DR	FPR	DR	FPR
Training sample (subset of Univ2007)						
Non-SKYPE	0.993	0.003	0.957	0.118	0.901	0.020
SKYPE	0.997	0.007	0.882	0.043	0.980	0.099
Univ2007 test data sets						
Non-SKYPE	0.993	0.004	0.957	0.117	0.901	0.019
SKYPE	0.996	0.007	0.883	0.043	0.981	0.099
Univ2010 test data sets						
Non-SKYPE	0.939	0.203	0.921	0.184	0.907	0.140
SKYPE	0.797	0.061	0.816	0.079	0.860	0.093

Table 9 Best model results on testing for the flow-based feature set for Skype detection (ITALY data sets)

	C5.0		AdaBoost		GP	
	DR	FPR	DR	FPR	DR	FPR
ITALY TCPE2X test data sets						
Non-SKYPE	N/A	0.014	N/A	1.00	N/A	0.293
SKYPE	0.986	N/A	0.000	N/A	0.707	N/A
ITALY UDPE2E test data sets						
Non-SKYPE	N/A	0.113	N/A	0.794	N/A	0.387
SKYPE	0.887	N/A	0.206	N/A	0.613	N/A
ITALY UDPE2O test data sets						
Non-SKYPE	N/A	0.470	N/A	0.976	N/A	0.065
SKYPE	0.530	N/A	0.024	N/A	0.935	N/A
ITALY UDPSIG test data sets						
Non-SKYPE	N/A	0.079	N/A	0.302	N/A	0.388
SKYPE	0.921	N/A	0.698	N/A	0.612	N/A

4.1.1 Discussion of Results

In this section, the robustness of the solutions generated automatically by the AdaBoost, GP, and C5.0 based classifiers were explored for identifying encrypted VoIP traffic, specifically Skype, in a given traffic trace. To do so, traffic traces from Dalhousie University, NIMS lab, WIDE (IPv6) and ITALY traces were employed. Furthermore, flow features were explored on these traces without using IP addresses, port numbers, or payload data.

The flow results demonstrate that the C5.0 classifier was the most consistent performer across all test and training conditions. This shows not only that the model which the C5.0 classifier learned during training was robust (generalized) enough to

Table 10 Best model results on testing for the flow-based feature set for Skype detection (NIMS2 data sets)

	C5.0		AdaBoost		GP	
	DR	FPR	DR	FPR	DR	FPR
NIMS2 GTALK2009 test data sets						
Non-SKYPE	0.787	N/A	0.999	N/A	0.751	N/A
SKYPE	N/A	0.213	N/A	0.001	N/A	0.249
NIMS2 PRIMUS2009 test data sets						
Non-SKYPE	1.00	N/A	1.00	N/A	0.820	N/A
SKYPE	N/A	0.000	N/A	0.000	N/A	0.180
NIMS2 ZFONE2009 test data sets						
Non-SKYPE	0.962	N/A	0.938	N/A	0.810	N/A
SKYPE	N/A	0.038	N/A	0.062	N/A	0.190

Table 11 Best model results on testing for the flow-based feature set for Skype detection (NIMS3 data sets)

	C5.0		AdaBoost		GP	
	DR	FPR	DR	FPR	DR	FPR
NIMS3 GTALK2010 test data sets						
Non-SKYPE	0.809	N/A	0.998	N/A	0.464	N/A
SKYPE	N/A	0.191	N/A	0.002	N/A	0.536
NIMS3 PRIMUS2010 test data sets						
Non-SKYPE	0.998	N/A	0.650	N/A	0.220	N/A
SKYPE	N/A	0.002	N/A	0.350	N/A	0.780
NIMS3 YAHOO2010 test data sets						
Non-SKYPE	0.927	N/A	0.931	N/A	0.581	N/A
SKYPE	N/A	0.073	N/A	0.069	N/A	0.419
NIMS3 RADIO2010 test data sets						
Non-SKYPE	0.997	N/A	0.997	N/A	0.997	N/A
SKYPE	N/A	0.003	N/A	0.003	N/A	0.003
NIMS3 TORRENT2010 test data sets						
Non-SKYPE	0.952	N/A	0.927	N/A	0.821	N/A
SKYPE	N/A	0.048	N/A	0.073	N/A	0.179
NIMS3 TV2010 test data sets						
Non-SKYPE	0.989	N/A	1.00	N/A	0.955	N/A
SKYPE	N/A	0.011	N/A	0.000	N/A	0.045
NIMS3 VPN2010 test data sets						
Non-SKYPE	0.987	N/A	1.00	N/A	0.595	N/A
SKYPE	N/A	0.013	N/A	0.000	N/A	0.405

Table 12 Best model results on testing for the flow-based feature set for Skype detection (IPv6 data sets)

	C5.0		AdaBoost		GP	
	DR	FPR	DR	FPR	DR	FPR
IPv6 2000 test data sets						
Non-SKYPE	0.999	N/A	1.00	N/A	0.136	N/A
SKYPE	N/A	0.001	N/A	0.000	N/A	0.864
IPv6 2009 test data sets						
Non-SKYPE	0.889	N/A	1.00	N/A	0.875	N/A
SKYPE	N/A	0.111	N/A	0.000	N/A	0.125

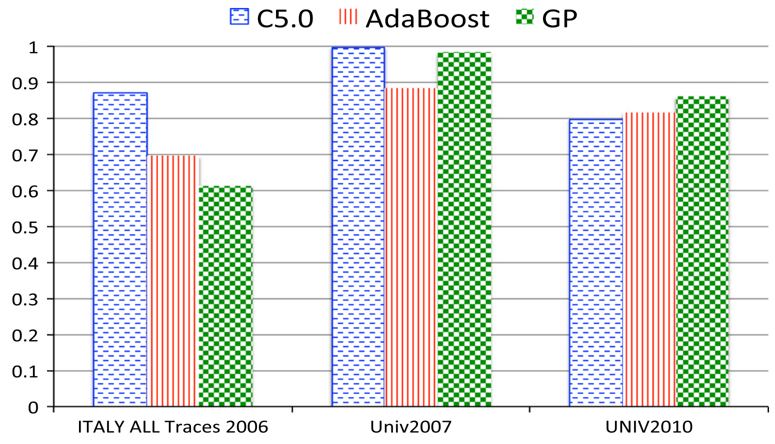


Fig. 6 The effect of time period in the DR for the three classifiers using a flow-based feature set for Skype detection on the test traces containing Skype traffic

be tested on real world network traces, but it also verified that achieving high detection and low FPRs were possible. In short, these results suggest that the flow-based classification system trained on data from one network can be employed to run on a different network and different time periods without new training.

Figures 6 and 7 list the performance of the three classifiers on traces that contain only Skype traffic from different time periods. The time periods range from 2006 to 2010. It is clear that the C5.0-based classifier had the best performance. The signatures found by C5.0 were able to achieve 87 % DR on traces that were captured a year earlier (the ITALY traces), and 86 % DR on traces that were captured 3 years later (the University traces). Even though Skype changed updates over the years, from version 2.5 in 2006 [63] to version 5.0 in 2010 [64], the C5.0-based classifier was able to identify Skype with a high DR and low FPR. We also think that the student/staff behaviour of using Skype did not change significantly from 2007 to 2010 at our University. Thus, the C5.0-based signatures can generalize well from one network to another and from different time periods as well as

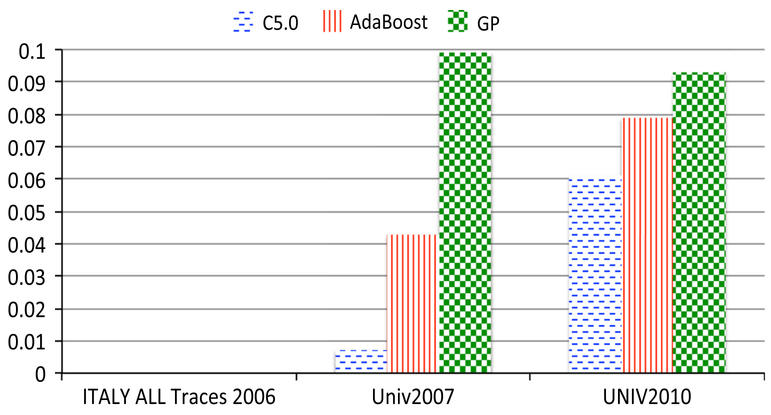


Fig. 7 The effect of time period in the FPR for the three classifiers using a flow-based feature set for Skype detection on test traces containing Skype traffic

different locations. These results imply that signatures based on flow features can adapt to the changing conditions of the applications and networks to some extent, and therefore, they can generalize well, i.e., they are robust. Furthermore, the C5.0 signatures were able to accurately classify VoIP Skype traffic that was captured 3 years later, which implied that the signatures could be valid for a period of 3 years. Hence, the updating of the signatures (retraining of the ML algorithms) could be performed at intervals of 3 years. Thus, cost and resources could be saved by applying such a guideline for signatures.

4.2 Evasion of the Robust Signatures

In this section, we examine the robustness/generalization of the proposed approach using ML algorithms with a flow based feature set by evaluating it against potential evasion attacks that could circumvent the traffic classifier. To the best of the authors knowledge, this is the first time the robustness/generalization of such an approach was investigated against evasion attacks as well as data from a different network and time period. To this end, we will first describe the data sets employed, then describe the experiments performed, and finally, discuss the results.

4.2.1 Altering Skype Traffic: Evasion Attacks

In this section, the performance of the signatures generated by C5.0, AdaBoost and GP against the padding of the Skype VoIP network traffic is investigated. To this end, we employed the same speech corpus audio files used in the generation of the NIMS Lab traces. The Speex encoder [65] was used to change the bit rate of the audio file from 8 Hz (the original bit rate of the audio file employed) to a range from 5 to 14 Hz as well as used a sound file convertor [66] to change the formatting type of the audio file. The audio formatting files employed were Macintosh sound formats (AIF and AIFF), a Sun Microsystems file format (AU), and MPEG-2 Audio Layer III (MP3). Table 13 lists the properties of the modified audio files.

Table 13 Statistical overview of altered-Skype audio files

Audio file	Duration (s)	Size in bytes	Number of flows
Original (8 Hz)	49	791 KB	2,396
Altering bitrate (5 Hz)	1:18	791 KB	3,188
Altering bitrate (6 Hz)	1:05	791 KB	2,863
Altering bitrate (7 Hz)	56	791 KB	2,324
Altering bitrate (9 Hz)	43	791 KB	3,494
Altering bitrate (10 Hz)	39	791 KB	3,605
Altering bitrate (11 Hz)	35	791 KB	2,478
Altering bitrate (12 Hz)	32	791 KB	2,821
Altering bitrate (13 Hz)	30	791 KB	3,235
Altering bitrate (14 Hz)	28	791 KB	3,123
Altering format (AIF)	49	791 KB	4,937
Altering format (AIFF)	49	791 KB	3,097
Altering format (AU)	49	8.7 MB	2,476
Altering format (MP3)	49	795 KB	2,041

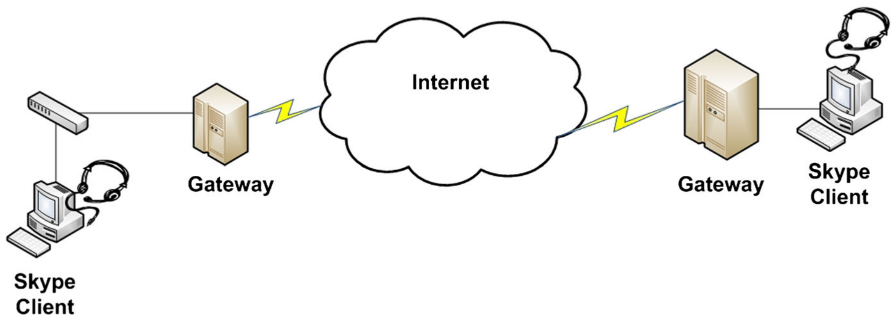


Fig. 8 Call setup for Skype evasion attacks

In these experiments, the impact of two different approaches for altering the audio on the transmission of the Skype VoIP calls were investigated. The network setup used was the same as with the NIMS2 Gtalk experiments in Sect. 3.5.3. A Skype client was installed on the Windows XP machines. The general call setup was the modified audio file played for ten minutes and then the output of the Windows media player was used as input for the Skype client using a microphone (Fig. 8). Wireshark was used for capturing the traffic on both ends of the Skype communication. These traffic traces (both altered and unaltered Skype traffic) were mixed with the lab traces described in Sects. 3.5.3 and 3.5.4.

Figure 9 plots the audio spectrum of four audio files. These files (from top to bottom of the figure) are the altered 6 Hz Bitrate audio file, the original audio file, the altered 9 Hz Bitrate audio file and the altered 14 Hz Bitrate audio file. The

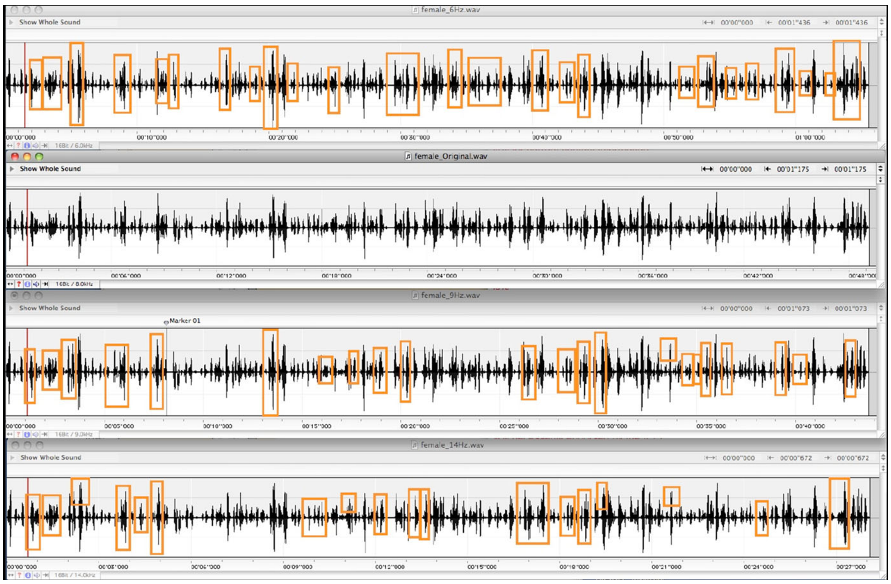


Fig. 9 Spectrum analyzer displays of audio signals

original audio file was an 8 Hz female voice audio file. The rectangle boxes on the altered signals indicate the differences between the altered data and the original data caused by the padding, the method used for the evasion attacks. As can be seen from these plots, the padding led to the production of different output, but it is very subtle. Furthermore, the altering of the original audio files changed the number of flows Skype generated when these audio files were employed in the Skype voice chat service (Table 13).

4.2.2 Evaluation of Evasion Attacks

The objective here is to evaluate the effectiveness of the C5.0, AdaBoost, and GP based classifiers (Flow-based models) trained on the Univ2007 training data set employed in Sect. 4.1 on both the original Skype traffic (not altered) traces and the evaded Skype traffic (altered) traces. As discussed earlier, both of these traces were generated in the NIMS lab. Table 14 shows the DR and FPR for C5.0, AdaBoost, and GP based classifiers both on the Skype original flows mixed with the NIMS traces and the Skype altered flows mixed with the NIMS traces. The reason the NIMS traces were used as the out-class was to see the performance of the signatures on a data set in which there were additional VoIP applications as well as other encrypted and non-encrypted applications. Moreover, the same network infrastructures were used to generate the NIMS traces and the Skype altered traffic.

Table 14 shows the results of the three ML on the Skype and Skype altered traffic. C5.0-based signatures show very promising performance for original Skype traffic with a ≈91 % DR and ≈5 % FPR for in-class traffic (Skype) and a ≈95 %

Table 14 C5.0 Robustness signatures results on the NIMS traces and Alter Skype traces

	C5.0		AdaBoost		GP	
	DR	FPR	DR	FPR	DR	FPR
Original Skype traces						
Non-SKYPE	0.953	0.087	0.941	0.087	0.767	0.063
SKYPE	0.913	0.047	0.913	0.059	0.937	0.233
Alter Skype traces						
Non-SKYPE	0.953	0.155	0.941	0.154	0.767	0.151
SKYPE	0.845	0.047	0.846	0.059	0.849	0.233

DR and ≈9 % FPR for out-class traffic (non-Skype). For Altered-Skype traffic, the C5.0-based classifier achieved an ≈85 % DR and ≈5 % FPR for in-class traffic (Altered-Skype) and a ≈95 % DR and ≈15 % FPR for out-class traffic (non-Skype). The AdaBoost-based classifier achieved an ≈85 % DR and ≈6 % FPR for in-class traffic (Altered-Skype) and a ≈94 % DR and ≈15 % FPR for out-class traffic (non-Skype). For original Skype traffic, the AdaBoost-based classifier achieved a ≈91 % DR and ≈6 % FPR for in-class traffic (Skype) and a ≈94 % DR and ≈9 % FPR for out-class traffic (non-Skype). The GP-based classifier achieved a ≈94 % DR and ≈23 % FPR for in-class traffic (Skype) and a ≈77 % DR and ≈6 % FPR for out-class traffic (non-Skype). For Altered-Skype traffic, GP achieved an ≈85 % DR and ≈23 % FPR for in-class traffic (Altered-Skype) and a ≈77 % DR and ≈15 % FPR for out-class traffic (non-Skype).

Table 15 lists the performance of the three ML-based classifiers on each of the flow data sets generated by altering the audio files. The lowest performance occurred when the audio file bit rate was changed from 8 to 9 Hz (TP ≈76 %). In terms of modifying the encoding format, the MP3 caused the lowest performance (≈82 % TP rate for the C5.0-based classifier). Specifically, the C5.0 achieved the highest performance on five of the Skype altered traces (5, 7, 12 Hz, AIFF and AU) and had a tie with AdaBoost for the best performance on the MP3 trace.

4.2.3 Discussion of Results

The modification of the audio files (Fig. 9) had a big effect on how Skype transmits the packets. The codecs were able to have an effect on the packet size and the duration required to transmit the packet. Hence, the characteristic of the flows changed, which makes them different from Skype flow traffic. With the success in padding the Skype payload by altering audio files to mimic the behaviour of other protocols, the performance of the signatures generated by C5.0 dropped by 7 % on average (Table 15). Therefore, the performance of the signatures generated by the ML algorithms has influenced by the audio and codecs type. Hence, the ML algorithms might be limited to adopt to the rapid changes of behaviour of the application or the network.

Table 15 The performance of the three classifiers on each of the altered-Skype traces

	C5.0		AdaBoost		GP	
	DR	FPR	DR	FPR	DR	FPR
5 Hz bitrate	0.88	0.00	0.87	0.00	0.84	0.00
6 Hz bitrate	0.78	0.00	0.83	0.00	0.81	0.00
7 Hz bitrate	0.82	0.00	0.81	0.00	0.77	0.00
9 Hz bitrate	0.76	0.00	0.75	0.00	0.77	0.00
10 Hz bitrate	0.89	0.00	0.81	0.00	0.89	0.00
11 Hz bitrate	0.86	0.00	0.85	0.00	0.89	0.00
12 Hz bitrate	0.84	0.00	0.82	0.00	0.83	0.00
13 Hz bitrate	0.82	0.00	0.90	0.00	0.86	0.00
14 Hz bitrate	0.82	0.00	0.91	0.00	0.92	0.00
AIF encoding	0.90	0.00	0.87	0.00	0.92	0.00
AIFF encoding	0.88	0.00	0.88	0.00	0.83	0.00
AU encoding	0.88	0.00	0.88	0.00	0.84	0.00
MP3 encoding	0.82	0.00	0.82	0.00	0.76	0.00

Previously published research [31, 32] in the field claimed that evasion attacks against ML-based classifiers and / or statistical features could succeed easily, resulting in performances below 50 % (random guessing). However, our experiments shown in this paper indicate that such classifiers are not as easy to evade as claimed before. Indeed, the performance of the traffic classifiers does drop against such attacks but these results indicate that well chosen training data sets and features can improve the generalization of such learning and data mining techniques even against evasion attacks. The results of this research suggest that the signatures generated by the C5.0 classifier do not become ineffective when faced with evasion attacks, i.e., altered Skype flows. Instead, they can still classify, albeit with a 7 % decrease in their performance. This suggests that the signatures are robust for classifying Skype traffic regardless of the effect of locations, time periods or padding payload packets to a certain limit.

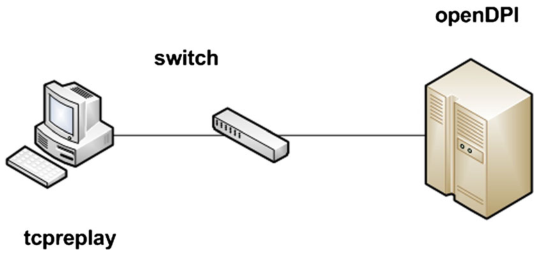
4.3 Evading Public Tools

In this section, we analyze the performance of the DPI tools on the traces discussed above.

4.3.1 Evading Attacks Against OpenDPI

The publicly available OpenDPI [67] is one of the common DPI tools used to classify traffic. Since OpenDPI requires the packet payload in order to classify the packets, the same traces used to evaluate the robustness of the ML algorithms against evasion attacks are employed in this section, too. Both the original Skype traffic generated in the NIMS lab and the altered-Skype traffic traces were used in order to evaluate the performance of the OpenDPI tool.

Fig. 10 OpenDPI testbed setup



The network testbed setup for this experiment (as shown in Fig. 10) is self explanatory. OpenDPI failed to classify the Altered-Skype traces with a ≈ 0 % DR and ≈ 99 % FPR. This is due to the fact that the signatures employed by the OpenDPI for classifying Skype are not effective on this data set. Also, these results indicate that there exists a need for alternative techniques (such as the one proposed in this research) for classifying this kind of traffic.

4.3.2 Evading Attacks Against Wireshark

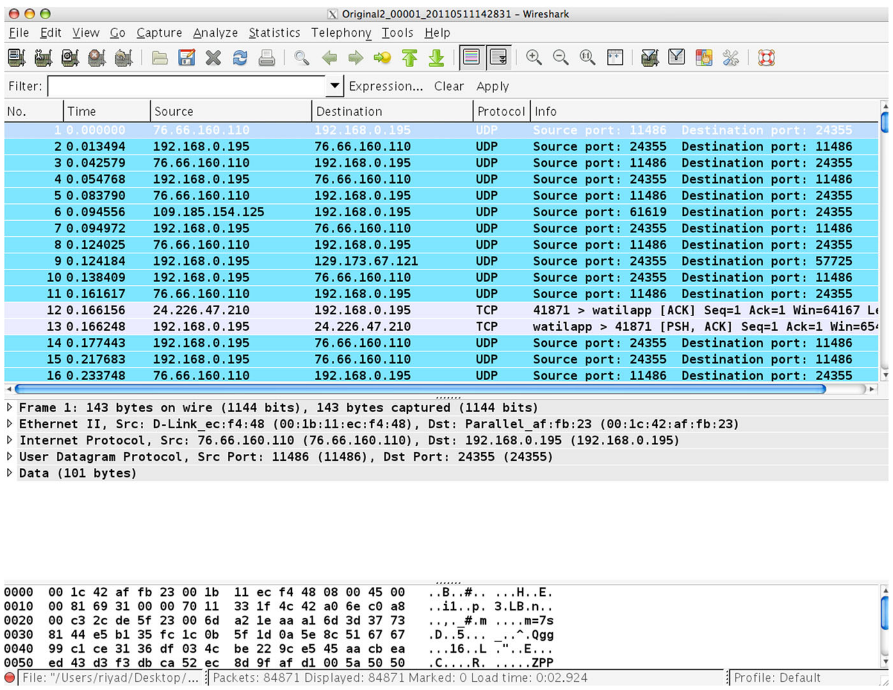
The purpose of this experiment is to show the effectiveness of Wireshark type traffic analyzers, which inspect all the headers (including the IP addresses and TCP/UDP port numbers) as well as the payload information, as traffic classification systems. Wireshark uses signatures based on port numbers and payload information to determine the type of traffic. Since Wireshark can be used as a network analysis tool to label traffic according to application type, Wireshark was also run on the Original Skype and Altered-Skype traffic. Figures 11 and 12 show the results in which Wireshark failed to detect any of Skype packets (Original or Altered).

4.3.3 Discussion

This section demonstrated how easy it is to evade public tools used for traffic classification that are based on the DPI method (OpenDPI) and the port number based method (Wireshark). The results of these experiments show the importance and the need of the proposed approach for finding robust signatures that are difficult to evade. To do so, it was necessary to evaluate the robustness/generalization of the solutions provided by the ML algorithms C5.0, GP, and AdaBoost in Skype traffic classification on unseen altered data by padding/morphing, i.e., evasion attacks. These experimental results suggest that the proposed approach provides over an 80 % DR on average for Skype VoIP traffic classification even if a user alters the characteristics of the Skype traffic maliciously. Our results show that the C5.0-based classifier (on average) has a better performance than the AdaBoost and the GP based classifiers.

4.4 Sensitivity Analysis of Configuration Parameters

This section discusses the effect of the parameters on the performance of the C5.0, since it is the top performer in most of our experiments. In the experiments we


Original2_00001_20110511142831 - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: Expression... Clear Apply

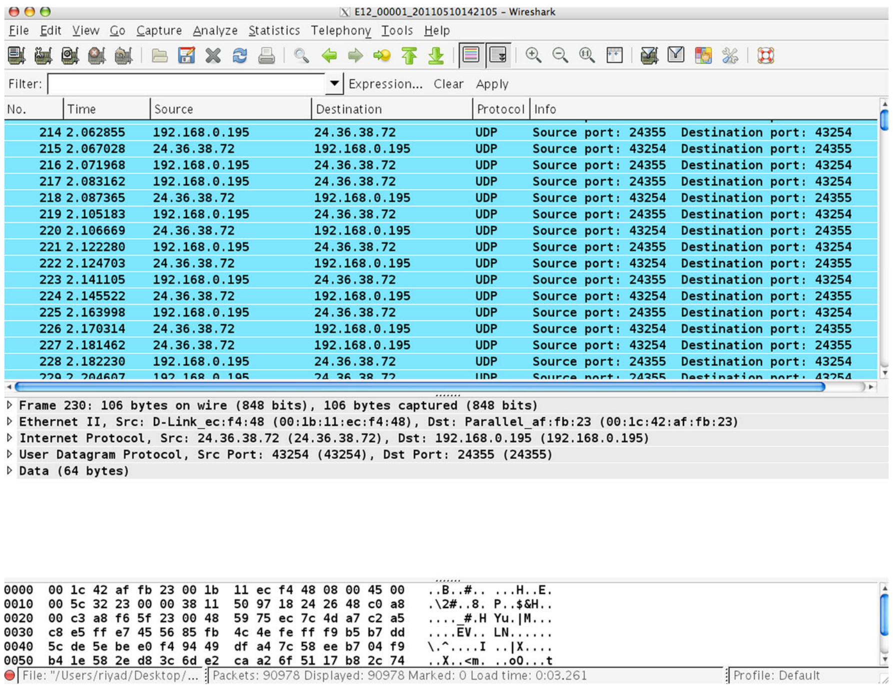
```
No.  Time      Source          Destination     Protocol Info
  1 0.000000  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355
  2 0.013494  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
  3 0.042579  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355
  4 0.054768  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
  5 0.083790  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355
  6 0.094556  109.185.154.125 192.168.0.195   UDP      Source port: 61619  Destination port: 24355
  7 0.094972  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
  8 0.124025  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355
  9 0.124184  192.168.0.195   129.173.67.121  UDP      Source port: 24355  Destination port: 57725
 10 0.138409  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
 11 0.161617  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355
 12 0.166156  24.226.47.210   192.168.0.195   TCP      41871 > watilapp [ACK] Seq=1 Ack=1 Win=64167 Le
 13 0.166248  192.168.0.195   24.226.47.210   TCP      watilapp > 41871 [PSH, ACK] Seq=1 Ack=1 Win=65
 14 0.177443  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
 15 0.217683  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
 16 0.233748  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355

▷ Frame 1: 143 bytes on wire (1144 bits), 143 bytes captured (1144 bits)
▷ Ethernet II, Src: D-Link_ec:f4:48 (00:1b:11:ec:f4:48), Dst: Parallel_af:fb:23 (00:1c:42:af:fb:23)
▷ Internet Protocol, Src: 76.66.160.110 (76.66.160.110), Dst: 192.168.0.195 (192.168.0.195)
▷ User Datagram Protocol, Src Port: 11486 (11486), Dst Port: 24355 (24355)
▷ Data (101 bytes)

0000  00 1c 42 af fb 23 00 1b  11 ec f4 48 08 00 45 00   ..B..#.. ...H..E.
0010  00 81 69 31 00 00 70 11  33 1f 4c 42 a0 6e c0 a8   ..i1..p. 3.LB.n..
0020  00 c3 2c de 5f 23 00 6d  a2 1e aa a1 6d 3d 37 73   ..,._#.m ....m=7s
0030  81 44 e5 b1 35 fc 1c 0b  5f 1d 0a 5e 8c 51 67 67   .D..5... _..^.Qgg
0040  99 c1 ce 31 36 df 03 4c  be 22 9c e5 45 aa cb ea   ...16..L ."..E...
0050  ed 43 d3 f3 db ca 52 ec  8d 9f af d1 00 5a 50 50   .C....R. .....ZPP
File: "/Users/riyad/Desktop/...  Packets: 84871 Displayed: 84871 Marked: 0 Load time: 0:02.924  Profile: Default
```

Fig. 11 Wireshark could not classify Skype original packets

presented up to now, the default parameters for the C5.0-based classifier have been used because the 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 C5.0 is the pruning/confidence factor, where this has a direct impact on the resulting model complexity. Model complexity is related to generalization and performance. Figures 13, 14, and 15 summarize the impact of varying the confidence factor (CF) on the DR, FPR, and the number of rules.[1] Essentially, as the CF increases (less pruning) the resulting C5.0 model becomes more specific; the FPR decreases and the number of rules increases. However, in this particular data set, there is no change in the DR. The best DR and FPR appear in the 5–45 CF interval. After this, any further improvement to the FPR is negligible. Unfortunately, this comes with a higher number of rules under C5.0 (more than 270 rules). The sweet spot in this configuration happens to correspond to the default parameterization for C5.0, Table 2.

C5.0 employs 187 rules for identifying Skype traffic and 215 rules (signatures) for non-Skype traffic. Figure 16 shows part of the C50 solution for detection Skype traffic based on the flow feature set. The C5.0 models/solutions for Skype include 5 features with more than 50 % usage, Table 16. Intuitively, what the C5.0 algorithm learned from the data makes sense. In order to correctly identify Skype traffic, the

[1] Skype detection task and flow features.


E12_00001_20110510142105 – Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

```
No.  Time      Source         Destination    Protocol Info
214 2.062855  192.168.0.195  24.36.38.72    UDP   Source port: 24355  Destination port: 43254
215 2.067028  24.36.38.72    192.168.0.195  UDP   Source port: 43254  Destination port: 24355
216 2.071968  192.168.0.195  24.36.38.72    UDP   Source port: 24355  Destination port: 43254
217 2.083162  192.168.0.195  24.36.38.72    UDP   Source port: 24355  Destination port: 43254
218 2.087365  24.36.38.72    192.168.0.195  UDP   Source port: 43254  Destination port: 24355
219 2.105183  192.168.0.195  24.36.38.72    UDP   Source port: 24355  Destination port: 43254
220 2.106669  24.36.38.72    192.168.0.195  UDP   Source port: 43254  Destination port: 24355
221 2.122280  192.168.0.195  24.36.38.72    UDP   Source port: 24355  Destination port: 43254
222 2.124703  24.36.38.72    192.168.0.195  UDP   Source port: 43254  Destination port: 24355
223 2.141105  192.168.0.195  24.36.38.72    UDP   Source port: 24355  Destination port: 43254
224 2.145522  24.36.38.72    192.168.0.195  UDP   Source port: 43254  Destination port: 24355
225 2.163998  192.168.0.195  24.36.38.72    UDP   Source port: 24355  Destination port: 43254
226 2.170314  24.36.38.72    192.168.0.195  UDP   Source port: 43254  Destination port: 24355
227 2.181462  24.36.38.72    192.168.0.195  UDP   Source port: 43254  Destination port: 24355
228 2.182230  192.168.0.195  24.36.38.72    UDP   Source port: 24355  Destination port: 43254
229 2.204607  192.168.0.195  24.36.38.72    UDP   Source port: 24355  Destination port: 43254

▷ Frame 230: 106 bytes on wire (848 bits), 106 bytes captured (848 bits)
▷ Ethernet II, Src: D-Link_ec:f4:48 (00:1b:11:ec:f4:48), Dst: Parallel_af:fb:23 (00:1c:42:af:fb:23)
▷ Internet Protocol, Src: 24.36.38.72 (24.36.38.72), Dst: 192.168.0.195 (192.168.0.195)
▷ User Datagram Protocol, Src Port: 43254 (43254), Dst Port: 24355 (24355)
▷ Data (64 bytes)

0000  00 1c 42 af fb 23 00 1b  11 ec f4 48 08 00 45 00   ..B..#.. ...H..E.
0010  00 5c 32 23 00 00 38 11  50 97 18 24 26 48 c0 a8   .\2#..8. P..$&H..
0020  00 c3 a8 f6 5f 23 00 48  59 75 ec 7c 4d a7 c2 a5   ...._#.H Yu.|M...
0030  c8 e5 ff e7 45 56 85 fb  4c 4e fe ff f9 b5 b7 dd   ....EV.. LN......
0040  5c de 5e be e0 f4 94 49  df a4 7c 58 ee b7 04 f9   \.^....I ..|X....
0050  b4 1e 58 2e d8 3c 6d e2  ca a2 6f 51 17 b8 2c 74   ..X..<m. ..oO...t

File: "/Users/riyad/Desktop/...  Packets: 90978 Displayed: 90978 Marked: 0 Load time: 0:03.261  Profile: Default
```

Fig. 12 Wireshark could not classify altered-Skype packets

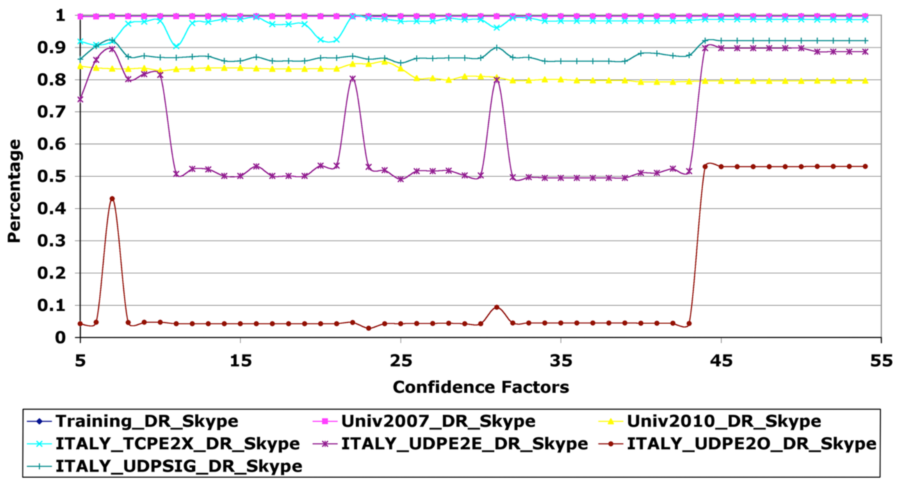


Fig. 13 Sensitivity of C5.0 DR for changing the confidence factor parameter

classifier naturally needs to explore both directions of a flow (forward and backward directions) since the VoIP Skype is an interactive application between two users. The features from forward and backward directions are identical, which are the minimum and the maximum packet lengths. These features reflect the

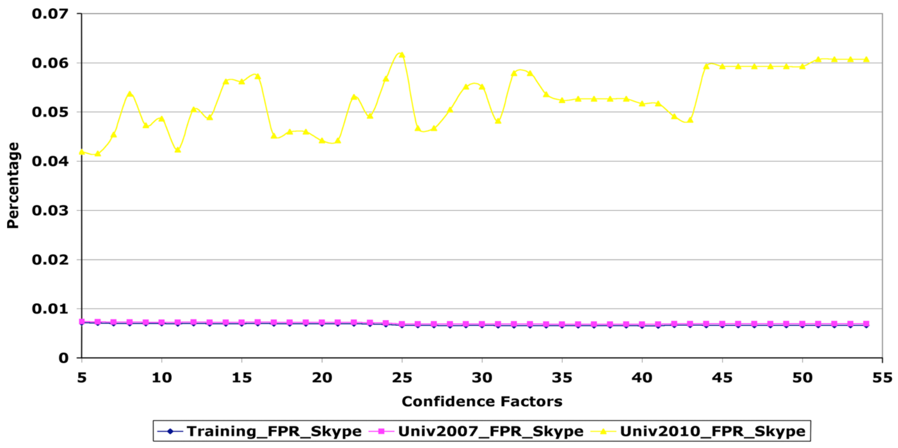


Fig. 14 Sensitivity of C5.0 FPR for changing the confidence factor parameter

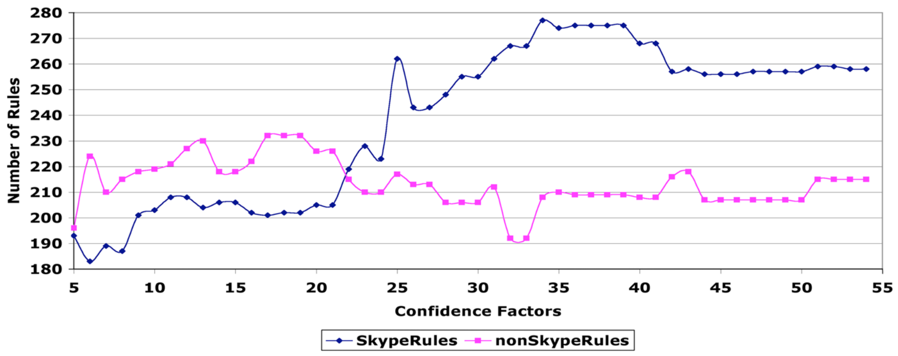


Fig. 15 Sensitivity of the number of C5.0 rules for changing the confidence factor parameter

communication aspect of both ends while the duration reflects on the quality of the communication. Hence, the C5.0 algorithm could identify the underlying pattern for the working mechanism of VoIP and build its model mostly from features such as min_bpktl, min_fpktl, max_bpktl, max_fpktl, total_bvolume, and duration. For example, the maximum length for a packet is effected by the length of the communication request made by the user, e.g., a user is asking a question: how are you, while the minimum length for a packet is effected by the communication of the responding user, e.g., user is giving an answer: fine. In summary, ML algorithms such as C5.0 select the most appropriate attributes (among the set given, Table 1) to build their classifier model/solution.

4.5 Limitations

No signature-based method in traffic classification can be perfect, given that there can always be some new (unseen) applications. Thus, the major challenge for traffic

Fig. 16 An example of a C5.0 solution for Skype detection based on the flow feature set

```
Rule 215: (879366/168737, lift 1.1)
        min_fpktl <= 139
        min_bpktl > 48
        -> class NOTSKYPE [0.808]

Rule 216: (9456/3, lift 3.5)
        std_fpktl <= 0
        max_bpktl <= 321
        std_bpktl > 80
        max_fiat <= 26620
        mean_biat > 34983
        mean_biat <= 330261
        -> class SKYPE [1.000]

Rule 217: (1801, lift 3.5)
        min_fpktl > 131
        duration <= 126901
        total_fvolume <= 132
        total_bvolume <= 63
        -> class SKYPE [0.999]

Rule 218: (798, lift 3.5)
        min_fiat > 0
        min_fiat <= 3
        -> class SKYPE [0.999]

Rule 219: (6044/3, lift 3.5)
        min_fpktl <= 94
        std_fpktl <= 2
        min_bpktl > 61
        min_bpktl <= 79
        max_bpktl > 244
        max_bpktl <= 298
        std_bpktl > 103
        -> class SKYPE [0.999]

Rule 220: (782, lift 3.5)
        min_fpktl > 34
        min_fpktl <= 50
        std_fpktl > 2
        min_bpktl > 46
        min_bpktl <= 79
        mean_bpktl > 164
        max_bpktl <= 595
        proto > 6
        total_fpackets <= 8
        total_bvolume > 359
        -> class SKYPE [0.999]
```

classification in general is evasion. All classification methods can be evaded. For example, a payload-based approach can be evaded by encrypting the packet payload and a port-based approach can be evaded by changing the port numbers dynamically. However, approaches based on flow statistics using packet size and inter-arrival time attributes are sensitive theoretically to altering these attributes. If attackers want to evade the proposed method they can modify the size of the packets in the entire connection by padding the packet payloads randomly. The accuracy of the proposed method might be decreased if those features that depend on packet size were modified from the application behaviour. However, it is not that easy to obfuscate application behaviour without presenting a large amount of overhead or changing the quality of the application. For example in the VoIP case, the amount of padding that can be implemented also has its limit. Too much padding may corrupt

Table 16 C5.0 feature usage

Percentage	Feature name
97	min_bpktl
93	min_fpktl
84	max_bpktl
66	max_fpktl
59	total_bvolume
56	duration
48	total_fvolume
39	min_biat
36	total_fpackets
33	std_bpktl
33	min_fiat
30	std_fpktl
23	proto
22	std_fiat
22	total_bpackets
18	mean_biat
17	mean_fpktl
15	mean_bpktl
8	std_biat
5	mean_fiat
2	max_fiat
1	max_biat

the voice service to the extent that the parties talking can no longer hear or understand each other. In this case, whether the application can evade the classifier or not is irrelevant given that it is no longer a useful application in terms of carrying voice data over the IP.

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 behaviour (such as new applications that are developed or old applications that change their behaviour). Indeed, in such cases, the signatures/models need to be updated by retraining the classifiers. That is why it is very important to conduct robustness analysis on such classifiers. For instance, in this research, C5.0 signatures have the best consistent performance in the robustness criteria and the signatures can classify Skype P2P VoIP traffic in a trace robustly if the characteristic of the flow features in the trace fall within the range described in Table 17.

Table 17 Ranges of values for each of the flow features for the signatures generated by C5.0

Feature name	Range
Duration	$value \leq 5{,}987{,}061$
fpackets	$value \leq 1{,}218$
fbytes	$value \leq 2{,}512$
bpackts	$value \leq 1{,}011$
bbytes	$value \leq 2{,}877$
min_fiat	$472{,}914 \leq value > 472{,}914$
mean_fiat	$47 \leq value > 420{,}485$
max_fiat	$47 \leq value > 26{,}620$
std_fiat	$67{,}009 \leq value > 6{,}202$
min_biat	$value \leq 569{,}517$
mean_biat	$value \leq 330{,}261$
max_biat	$value \leq 20{,}162$
std_biat	$value \leq 85{,}003$
min_fpkt	$value > 30$
mean_fpkt	$512 \leq value > 47$
max_fpkt	$344 \leq value > 47$
std_fpkt	$value \leq 548$
min_bpkt	$870 \leq value > 30$
mean_bpkt	$1{,}299 \leq value > 38$
max_bpkt	$1{,}341 \leq value > 42$
std_bpkt	$633 < value > 587$

5 Conclusion and Future Directions

In this research paper, the main goal was to investigate the robustness of the models/ signatures generated automatically by an ML-based approach—specifically C5.0, GP, and AdaBoost—for distinguishing encrypted VoIP traffic (namely Skype) from other traffic in a given traffic trace. In this context, the robustness of a classifier means that the classifier is trained on a data set from one network traffic trace but tested on

1. unseen data from different locations/networks;
2. unseen data from different time periods;
3. unseen data that is altered by padding/morphing (evasion attacks)

To explore this, three supervised ML algorithms, C5.0, AdaBoost, and GP, were employed. It should be noted here that these learning algorithms are not opaque. In other words, their solutions can be analyzed to provide signatures (rules) that can be understood easily by human experts. Moreover, these learning algorithms are well known for selecting the most appropriate features from a given set for a given task.

In this research, the C5.0-based classification approach performed the best on the given data sets. In the best case scenario, the C5.0-based classifier achieved a

99.6 % DR with 0.7 %FPR (when trained on one network but tested on another) in detecting Skype traffic. 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. Additionally, the results of the evasion attack traces based on morphing techniques in Sect. 4.2 showed that C5.0 provided over an 80 % DR on average for Skype VoIP traffic classification on maliciously altered Skype traffic.

With regard to evasion attacks, the signatures generated by the C5.0-based classifier from a statistical feature set and a well chosen training data set were proven to be robust and not easy to evade. The C5.0-based classifier achieved a ≈91 % DR and ≈5 % FPR on the Original Skype flows and an ≈85 % DR and ≈5 % FPR on the Altered-Skype flows. On the other hand, when open source classifiers such as an openDPI and open source network analyzers such as Wireshark employed on the same evasion attack data sets, they were easily evaded (circumvented) by the Altered-Skype application.

In summary, the main contribution of this research paper is the empirical evaluation of the robustness by developing a classification system that is able to generate robust signatures automatically for classifying encrypted VoIP traffic, specifically Skype traffic, on traffic traces from different time periods, locations, and networks infrastructures as well as against evasion attacks.

Given the results obtained in this research paper, we would like to apply our approach against other evasion attacks and for classifying other encrypted traffic.

Acknowledgments This research is supported by a Natural Science and Engineering Research Council of Canada (NSERC) grant. This work is conducted as part of the Dalhousie NIMS Lab at http://projects.cs.dal.ca/projectx/.

References

1. IANA, Internet assigned numbers authority, http://www.iana.org/assignments/port-number (last Accessed Oct 2009)
2. Moore, A.W., Papagiannaki, K.: Toward the accurate identification of network applications. In: Passive and Active Network Measurement: Proceedings of the Passive & Active Measurement Workshop, pp. 41–54 (2005)
3. Madhukar, A., Williamson, C.: A longitudinal study of p2p traffic classification. In: Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. MASCOTS 2006. 14th IEEE International Symposium on, pp. 179–188 (2006)
4. Sen, S., Spatscheck, O., Wang, D.: Accurate, scalable in-network identification of p2p traffic using application signatures. In: WWW '04: Proceedings of the 13th International Conference on World Wide Web, pp. 512–521. ACM, New York, NY, USA (2004)
5. Erman, J., Arlitt, M., Mahanti, A.: Traffic classification using clustering algorithms. In: MineNet '06: Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data, pp. 281–286. ACM Press, New York, NY, USA (2006)
6. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINC: multilevel traffic classification in the dark. In: SIGCOMM '05: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 229–240. ACM Press, New York, NY, USA (2005)
7. Alshammari, R., Zincir-Heywood, A.N.: Can encrypted traffic be identified without port numbers, IP addresses and payload inspection? Comput. Netw. **55**(6), 1326–1350 (2011)
8. Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A., Salamatian, K.: Traffic classification on the fly. SIGCOMM Comput. Commun. Rev. **36**(2), 23–26 (2006)

9. Moore, A.W., Zuev, D.: 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, pp. 50–60. ACM Press, New York, NY, USA (2005)
10. Alshammari, R., Zincir-Heywood, A.N.: A flow based approach for ssh traffic detection. In: Proceedings of the IEEE International Conference on System, Man and Cybernetics—SMC'2007 (2007)
11. Alshammari, R., Zincir-Heywood, A.N.: Investigating two different approaches for encrypted traffic classification. In: PST '08: Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust, pp. 156–166. IEEE Computer Society, Washington, DC, USA (2008)
12. Alshammari, R., Zincir-Heywood, N.: Generalization of signatures for ssh encrypted traffic identification. In: Computational Intelligence in Cyber Security. CICS '09. IEEE Symposium on, pp. 167–174 (2009)
13. Early, J., Brodley, C., Rosenberg, C.: Behavioral authentication of server flows. In: Proceedings of the 19th Annual Computer Security Applications Conference, pp. 46–55 (2003)
14. Haffner, P., Sen, S., Spatscheck, O., Wang, D.: ACAS: automated construction of application signatures. In: MineNet '05: Proceeding of the 2005 ACM SIGCOMM Workshop on Mining Network Data, pp. 197–202. ACM Press, New York, NY, USA (2005)
15. Montigny-Leboeuf, A.D.: Flow Attributes for Use in Traffic Characterization, CRC Technical Note No. CRC-TN-2005-003, Feb 2005.
16. Wright, C., Monrose, F., Masson, G.M.: HMM profiles for network traffic classification. In: VizSEC/DMSEC '04: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, pp. 9–15. ACM Press, New York, NY, USA (2004)
17. Williams, N., Zander, S., Armitage, G.: A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. SIGCOMM Comput. Commun. Rev. **36**(5), 5–16 (2006)
18. Pise, N., Kulkarni, P.: A survey of semi-supervised learning methods. In: Computational Intelligence and Security. CIS '08. International Conference on, vol. 2, pp. 30–34 (2008)
19. Alshammari, R.: Automatically classifying encrypted network traffic: a case study of ssh. Mater thesis, Dalhousie University, NS, Canada, 133 pp. (2008)
20. Quinlan, J.: See5-comparison, http://www.rulequest.com/see5-comparison.html (last Accessed Feb 2011)
21. Callado, A., Kelner, J., Sadok, D.: Alberto Kamienski C, Fernandes S.: Better network traffic identification through the independent combination of techniques. J. Netw. Comput. Appl. **33**(4), 433–446 (2010)
22. Nguyen, T., Armitage, G.: A survey of techniques for internet traffic classification using machine learning. Commun. Surv. Tutor. IEEE **10**(4), 56–76 (2008)
23. Callado, A., Kamienski, C., Szabo, G., Gero, B., Kelner, J., Fernandes, S., Sadok, D.: A survey on internet traffic identification. Commun. Surv. Tutor. IEEE **11**(3), 37–52 (2009)
24. Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., Tofanelli, P.: Revealing skype traffic: when randomness plays with you. SIGCOMM Comput. Commun. Rev. **37**(4), 37–48 (2007)
25. Freire, E., Ziviani, A., Salles, R.: Detecting skype flows in web traffic. In: Network Operations and Management Symposium. NOMS 2008, pp. 89–96. IEEE (2008)
26. Este, A., Gringoli, F., Salgarelli, L.: Support vector machines for TCP traffic classification. Comput. Netw. **53**(14), 2476–2490 (2009)
27. Erman, J., Mahanti, A., Arlitt, M., Cohen, I., Williamson, C.: Offline/realtime traffic classification using semi-supervised learning. Perform. Eval. **64**, 1194–1213 (2007)
28. Bacquet, C., Gumus, K., Tizer, D., Zincir-Heywood, A., Heywood, M.I.: A comparison of unsupervised learning techniques for encrypted traffic identification. J. Inf. Assur. Secur. **5**, 464–472 (2010)
29. Iliofotou, M., Kim, H.C., Faloutsos, M., Mitzenmacher, M., Pappu, P., Varghese, G.: Graption: a graph-based p2p traffic classification framework for the internet backbone. Comput. Netw. **55**(8), 1909–1920 (2011)
30. Park, J., Tyan, H.-R., Kuo, C.-C.: Ga-based internet traffic classification technique for QoS provisioning. In: Intelligent Information Hiding and Multimedia Signal Processing. IIH-MSP '06. International Conference on, pp. 251–254 (2006)
31. Hu, Y., Chiu, D.-M., Lui, J.C.S.: Profiling and identification of p2p traffic. Comput. Netw. **53**(6), 849–863 (2009)

32. Wright, C.V., Coull, S.E., Monrose, F.: Traffic morphing: an efficient defense against statistical traffic analysis. In: Proceedings of the Network and Distributed Security Symposium—NDSS '09 (2009)
33. Wright, C.V., Ballard, L., Monrose, F., Masson, G.M.: Language identification of encrypted VoIP traffic: Alejandra y roberto or alice and bob? In: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, pp. 4:1–4:12. USENIX Association, Berkeley, CA, USA (2007)
34. Liberatore, M., Levine, B.N.: Inferring the source of encrypted http connections. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, pp. 255–263. ACM, New York, NY, USA (2006)
35. Skype, http://www.skype.com/useskype/
36. Baset, S.A., Schulzrinne, H.G.: An analysis of the skype peer-to-peer internet telephony protocol. In: INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pp. 1–11 (2006)
37. Bonfiglio, D., Mellia, M., Meo, M., Ritacca, N., Rossi, D.: Tracking down skype traffic. In: INFOCOM 2008. The 27th Conference on Computer Communications, pp. 261–265. IEEE (2008)
38. De Cicco, L., Mascolo, S., Palmisano, V.: Skype video responsiveness to bandwidth variations. In: Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '08), pp. 81–86. ACM, New York, NY, USA (2008)
39. Barbosa, R., Callado, A., Kamienski, C., Fernandes, S., Mariz, D., Sadok, D.: Performance evaluation of P2P VoIP application. In: Proceedings of the 17th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '07), IL, USA (2007)
40. IETF, http://www3.ietf.org/proceedings/97apr/97apr-final/xrtftr70.htm
41. NetMate, http://www.ip-measurement.org/tools/netmate/
42. Arndt, D.: How to calculating flow statistics using netmate, http://dan.arndt.ca/nims/calculating-flow-statistics-using-netmate/ (last Accessed Sept 2011)
43. Quinlan, J.: see5-info, http://www.rulequest.com/see5-info.html (last Accessed July 2010)
44. Alpaydin, E.: Introduction to Machine Learning. MIT Press, Cambridge, MA (2004)
45. Lichodzijewski, P., Heywood, M.I.: Managing team-based problem solving with symbiotic bid-based genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 363–370 (2008)
46. de Jong, E.: A monotonic archive for pareto-coevolution. Evol. Comput. **15**(1), 61–93 (2007)
47. T. U. of Waikato, WEKA software, http://www.cs.waikato.ac.nz/ml/weka/
48. SBB-GP, Symbiotic bid-based (sbb) paradigm, http://www.cs.dal.ca/mheywood/Code/SBB/SCM.9.r20081212.tar.gz (last Accessed March 2008)
49. PacketShaper, http://www.packeteer.com/products/p-acketshaper/ (last Accessed March 2008). CalladoBetter2010
50. Traces, S.: Telecommunication networks group—politecnico ditorino, http://tstat.tlc.polito.it/traces-skype.shtml (last Accessed August 2009)
51. Alshamamri, R.: Downloading the NIMS data sets, http://web.cs.dal.ca/riyad/Site/Download.html (last Accessed Sept 2011)
52. Wireshark, http://www.wireshark.org/ (last Accessed Sept 2008)
53. Peeker, N.: Netpeeker, http://www.net-peekerCalladoBetter2010.com (last Accessed Oct 2009)
54. Signalogic, Speech codec wav samples, http://www.signalogic.com/index.pl?page=codec_samples (last Accessed Oct 2009)
55. Zimmermann, P.: The Zfone project, http://zfoneproject.com/ (last Accessed Oct 2009)
56. Zimmermann, E.P., Johnston, A., Callas, J.: Zrtp: media path key agreement for secure rtp, http://tools.ietf.org/html/draft-zimmermann-avt-zrtp-17 (2010)
57. P. T. C. Inc, Primus softphone client, http://www.primus.ca/en/residential/talkbroadband/talkBroadband-softphone.htm (last Accessed Oct 2009)
58. ETSI, Digital cellular telecommunications system (phase 2+), general packet radio service (gprs), overall description of the gprs radio interface, stage 2 (gsm 03.64, version 7.0.0, release 1999)
59. BirdsSoft, Vpn-x, http://birdssoft.com/ (last Accessed March 2011)
60. Kent, S., Atkinson, R.: Security architecture for the internet protocol, http://www.ietf.org/rfc/rfc2401.txt (1998)
61. MAWI, Mawi working group traffic archive, http://tracer.csl.sony.co.jp/mawi/
62. Fink, R., Hinden, R.: 6bone (IPv6 testing address allocation), http://tools.ietf.org/html/rfc3701 (2004)

63. Ehlert, S., Petgang, S., Magedanz, T., Sisalem, D.: Analysis and signature of Skype VoIP session traffic. In: CIIT 2006: 4th IASTED International Conference on Communications, Internet, and Information Technology, pp. 83–89 (2006)
64. Skype, Skype garage, http://blogs.skype.com/garage/windows/ (last Accessed Sept 2011)
65. Valin, J.-M., Montgomery, C.: Improved noise weighting in CELP coding of speech—applying the Vorbis psychoacoustic model to speex, http://www.speex.org (2006)
66. N. software, Switch audio converter for mac, http://www.nch.com.au/switch/index.html (last Accessed May 2011)
67. OpenDPI, the open source version of ipoque's dpi engine, http://www.opendpi.org/ (last Accessed April 2011)

Dr. Riyad Alshammari obtained his B.S. Degree in Computer Science in 2005. In 2008, he attained his Masters of Computer Science and Ph.D. in Computer Science in 2012, all from Dalhousie University, Halifax, NS, Canada. In 2007, he was a Research Assistant and at the same time Teaching Assistant at the Faculty of Computer Science, University of Dalhousie Halifax, Canada. In 2012, he joined King Saud Bin Abdulaziz University for Health Sciences as Adjunct Instructor at the College of Public Health and Health Informatics for one semester after which he was credentialed as Assistant Professor, Department of Health Informatics, KSAU-HS. In July 2013, he has been appointed as an adjunct faculty at the Faculty of Computer Science at Dalhousie University, Canada.

Dr. A. Nur Zincir-Heywood is currently a full Professor of Computer Science at Dalhousie University, Canada. She received her Ph.D. in 1998 in Computer Science and Engineering from Ege University, Turkey. Prior to moving to Dalhousie in 2000, Dr. Zincir-Heywood had been a researcher at Sussex University, UK and Karlsruhe University, Germany as well as working as an instructor at the Internet Society Network Management workshops. She has published over 150 papers on network operations, security and web centric systems related research. She has substantial experience of industrial research in systems security and network operations related topics with Swisscom, Palomino, Genieknows, Spielo and Public Safety Canada. She is currently a PI on different research grants from Mitacs and NSERC. Her research interests include network operations, computer and network security, information extraction and computational intelligence. She is a member of the IEEE and the ACM.