A Preliminary Investigation of Skype Traffic Classification Using a Minimalist Feature Set

Duffy Angevine, A. Nur Zincir-Heywood
Dalhousie University, Faculty of Computer Science, Halifax, NS, Canada
{angevine,zincir}@cs.dal.ca

Abstract

In this work, AdaBoost and C4.5, are employed for classifying Skype direct (UDP and TCP) communications from traffic log files. Pre-processing is applied to the traffic data to express it as flows, which is later converted into a descriptive feature set. The aforementioned algorithms are then evaluated on this feature set. Results show that a 98% detection rate with 6% false positive rate for UDP based Skype and a 94% detection rate with 4% false positive rate for TCP based Skype is possible to achieve.

1. Introduction

In this work, our objective is to explore the limits of employing machine learning algorithms namely, C4.5 and AdaBoost, in order to classify Skype traffic without using the payload, IP addresses and port information from a given traffic file. By doing so, we aim to develop a framework where privacy concerns of users are respected but also an important task of network management, i.e. accurate identification of network traffic, is also achieved. Given that Skype is one of most popular VoIP applications used on the Net, it is important to identify it accurately for network engineering and management tasks. Moreover, being an encrypted application and tunneling itself through different ports, identifying Skype becomes even more challenging.

Traditionally, one approach to classifying applications is to inspect the payload of every packet. However, this is not realistic in practice. Firstly, there are privacy concerns with examining user data. Secondly, there is a high computational and storage overhead when attempting to study every packet that traverses a link; especially given the wide spread dependency on high-speed links. Thirdly, there are applications such as Skype, which has encrypted

Given the above, another approach to classifying traffic is using well-known TCP/UDP port numbers. This solves the issues regarding privacy concerns as well as the requirement for a high computational and storage overhead. However this approach has become increasingly inaccurate, mostly because applications use nonstandard ports to avoid detection, to by-pass firewalls or circumvent operating systems restrictions. Thus, other techniques are needed to increase the accuracy of network traffic identification.

One possibility is to identify specific features of the application traffic and use these to guide the traffic classification. Recent research in this area focuses on the identification of efficient and effective classifiers. Different research groups have employed various classification techniques such as Hidden Markov models, Naive Bayesian models, or AdaBoost, to this problem [3 - 6]. On the other hand, the limitations of port-based and payload-based analysis have motivated the use of transport/flow layer statistics for traffic classification [7 - 9]. These techniques rely on the fact that different applications have distinct behavior patters on the network. However, in general all these efforts show that even though it is easier to apply such techniques to well known application traffic such as HTTP (Hyper Text Transfer Protocol) and SMTP (Simple Mail Transfer Protocol), more work is needed to identify encrypted traffic such as Skype, which presents a unique challenge in terms of identification due to several key factors. These factors include using dynamic port allocation, payload encryption and various means of tunneling its service via different ports and protocols in order to evade detection and bypass firewalls. The remainder of the paper is organized as follows. An overview of Skype is given in section 2. Section 3 discusses the background research. Machine learning algorithms are presented in section 4. The methodology is described in sections 5 and experimental results are given in section 6. Finally, conclusions are drawn and future work is discussed in Section 7.

2. Skype Overview

Skype is a proprietary P2P(Peer to Peer) VoIP network. It was founded by Niklas Zennström and Janus Friis, who are also founders of Kazaa, a file sharing system. Skype is widely known for its broad range of features, including free voice and video conferencing, and its ability to use P2P technology to overcome common firewall and NAT problems [10]. Skype users can speak to other Skype users for free, call traditional telephone numbers for a fee (SkypeOut), receive calls from traditional phones (SkypeIn), and receive voicemail messages. It is a versatile method of synchronous and asynchronous communication.

Based on the information that Skype has provided on its website (http://www.skype.com) as well the work in [11,

12], it is apparent that from the outset Skype was designed to be not only 'secure' but also resilient to detection and blocking methods. While the encryption aspects of Skype are best described in [11, 12], we believe that the fact that the traffic is encrypted should aid the process of detection. Since VoIP is subject to delay and jitter on the network, we can assume that the ciphers are run on very small block sizes. This assumption is grounded in the rational that larger blocks will require more time to encrypt and decrypt, and furthermore when set via TCP, larger block sizes will be a detriment if the packet is lost during transmission or corrupted since the whole process will need to be repeated.

3. Detecting Skype

Most of the research existing in the literature on Skype reflects the experience of different research teams when building LibPcap tools. The common goal has been to observe and report on how Skype reacts under different modes of transmission [14]. On the other hand, most commercial products [15 - 20] claim to block/detect Skype via several means. For example, some block the communication to the login server of Skype (ui.skype.com) or some maintain a list of Skype Super Nodes and block the communication to these nodes. Others such as Cisco IOS [19] can limit its functionality by enforcing quality of service conditions, which would affect all other applications.

In this work, our objective is to detect Skype traffic automatically without using *a priori* information such as Skype login server information, super Node lists or port and IP addresses. This is very different from the existing research work and commercial products in the market. In order to achieve our objective, we divide the problem of classifying Skype traffic into two phases: (i) Skype UDP traffic; (ii) Skype TCP traffic.

4. Machine Learning

AdaBoost:

AdaBoost, an acronym for Adaptive Boosting, was developed by Yoav Freund and Robert Schapire [24]. It is a meta-algorithm, which means constructing a strong classifier as a linear combination of weak classifiers. It produces a sequence of gradually more complex classifiers in order to improve the overall performance by building a strong classifier from several weak classifiers.

These simple weak classifiers are called decision stumps. They examine the feature set and return a decision tree with two leaves. The leaves of the tree are used for binary classification and the root node evaluates the value of only one feature. Thus, each decision stump will return either +1 if the object is in class, or –1 if it is out class.

Once this process has been completed the resulting structure that AdaBoost returns is the final (strong) classifier with a weighted majority vote of T weak classifiers. This is defined to be the sequence of decision stumps that can best classify the training set with the given features. A more detailed explanation of AdaBoost can be found in [22].

C4.5:

C4.5 is a decision tree based classification algorithm. A decision tree is a hierarchical data structure for implementing a divide-and-conquer strategy. It is an efficient non-parametric method that can be used both for classification and regression. In non-parametric models, the input space is divided into local regions defined by a distance metric. In a decision tree, the local region is identified in a sequence of recursive splits in smaller number of steps. A decision tree is composed of internal decision nodes and terminal leaves. Each node m implements a test function $f_m(x)$ with discrete outcomes labeling the branches. This process starts at the root and is repeated until a leaf node is hit. The value of a leaf constitutes the output. In the case of a decision tree for classification, the goodness of a split is quantified by an impurity measure. A split is pure if for all branches, for all instances choosing a branch belongs to the same class after the split. One possible function to measure impurity is entropy, equation 1 [22].

$$I_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$$

If the split is not pure, then the instances should be split to decrease impurity, and there are multiple possible attributes on which a split can be done. Indeed, this is locally optimal, hence has no guarantee on finding the smallest decision tree. In this case, the total impurity after the split can be measured by equation 2 [22]. A more detailed explanation of the algorithm can be found in [22].

$$I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p_{mj}^i \log_2 p_{mj}^i$$

5. Methodology

As discussed earlier, we divided the problem of classifying Skype traffic into two phases. In the first phase, we develop a classifier for the automatic recognition of Skype Direct UDP traffic. In the second phase, we develop a classifier for the automatic recognition of Skype Direct TCP traffic. In all cases, we employ AdaBoost and C4.5 in order to identify the best fitting solution to the problem. The reason we employ these two machine learning algorithms is because they have been shown to be good classifiers in the previous literature on network traffic detection [23].

Data Collection:

To this end Dalhousie University Computing and Information Services Department (UCIS) provided us

with labeled network traffic. This data set consists of network traffic from the campus network during a continuous eight-hour period in January 2007. This traffic that was logged by UCIS, was aggregated by PacketShaper in both directions, going on and off campus.

The captured traffic contained at most the first 128 bytes of a packet, which allowed for the full IP header, as well as the TCP header if the options field was included. From this basic starting point, the UCIS department then proceeded to anonymize each packet, first by altering the IP address fields, then further truncating the packet so that no payload data was included. Other alterations to the packets include that checksums were reset to prevent attempts to guess the data.

Rather than classifying the data on a packet-by-packet inspection, which involves the complications of temporal elements into the packet stream, it was decided that the captured traffic traces be converted into bi-directional communication flows. By doing so, we aim to identify the potential unique patterns in Skype traffic flows. To this end, we developed our own flow tool set.

Since our objective is to establish results based on training on a partition of the UCIS data set, and then validating the results on unseen partitions of differing traffic percentages, it was decided to randomly partition the UCIS data set into different samples. These random partitions include the following characteristics:

- Contain at most 50% Skype traffic in the training partition of the data
- Contain anywhere between 15% to 50% Skype traffic in the testing partition of the data
- Contain no single packet flows.

It should be noted here that each sample used for training and/or testing in Table I contains 1000 flows. In this work, in addition to Skype flow, UDP sample data sets include applications such as Ventrilo (a VoIP software), traceroute, TACACS, SIP and SunRPC flows. On the other hand, TCP sample data sets include http, BitTorent, IRC, SIP and SunRPC application flows. In both cases, SIP and SunRPC flows are only included in the data sets with 15% Skype traffic.

Feature Selection:

Based on the above assumptions, we selected the following feature set to represent a flow to our system:

- Duration
- Number of bytes in the forward direction per second
- Number of packets in the forward direction per second
- Number of bytes in the backward direction per second
- Number of packets in the backward direction per second
- Number of ACK packets
- Protocol ID
- Minimum Packet length observed (not counting ACK, RST, SYN packets)
- Maximum packet length observed

These features are simple and well understood within the networking community. They represent reasonable benchmark feature sets to which more complex features might be added in the future. Other features such as IP addresses and source/destination port numbers are excluded from the data sets to ensure that the results are not dependent on such biased features or produce a trivial solution.

6. Results

In this section, we are going to present the results of the AdaBoost and C4.5 classifiers on our data sets using detection rate (DR) and false positive rate (FP). A high detection rate and a low false positive rate would be the desired outcomes. They are calculated as follows:

DR = 1 – (#FalseNegatives)/TotalNumberSkypeFlows
FP = #FalsePositives/TotalNumberNon-SkypeFlows

where False Negative means Skype traffic classified as non-Skype traffic and False Positive means non-Skype traffic classified as Skype traffic.

Once the aforementioned feature set is prepared for the data sets, then classifiers are trained on the data using Weka [21], which is an open source tool for data mining tasks, with its default parameters to run C4.5 and AdaBoost. Table 1 lists the results for the two machine learning algorithms.

Table 1: Results of the Classifiers

	AdaBoost		C4.5	
	DR	**FP**	**DR**	**FP**
UDP Training				
Skype	0.97	0.26	0.99	0.02
Non-Skype	0.74	0.03	0.98	0.01
UDP Testing (50% Skype)				
Skype	0.98	0.26	0.98	0.02
Non-Skype	0.74	0.02	0.98	0.02
UDP Testing (15% Skype)				
Skype	0.97	0.26	0.98	0.06
Non-Skype	0.74	0.03	0.94	0.02
TCP Training				
Skype	0.97	0.11	0.97	0.01
Non-Skype	0.89	0.03	0.99	0.03
TCP Testing (50% Skype)				
Skype	0.97	0.11	0.94	0.01
Non-Skype	0.89	0.03	0.99	0.06
TCP Testing (15% Skype)				
Skype	0.94	0.14	0.94	0.04
Non-Skype	0.86	0.06	0.96	0.06

As the above results show both algorithms achieve minimum 94% detection rate on the test data sets for TCP based Direct Skype traffic. On the other hand, C4.5 achieves minimum 98% detection rate on UDP based

direct Skype traffic, whereas AdaBoost achieves 97% detection rate under the same conditions. However differences between the performances of the two algorithms get more obvious when we look at the false positive rates for Skype traffic. In this case, we observe that AdaBoost misclassifies non-Skype traffic as Skype ~25% of the time under the UDP protocol and ~10% of the time under the TCP protocol. However, false positive rates given by C4.5 never go above 6%, which makes it a more desirable classifier for these tasks.

Analysis of Results:

The results indicate that C4.5 performed the best, considering the false positives and detection rates for both classes. As such, since C4.5 uses a form of information gain, we can analyze which features provided the most information to identify Skype accurately.

Figure 1 and Figure 2, show the fully pruned decision trees for the TCP and UDP based C4.5 classifiers, respectively. As these figures show, the packet size feature is a very good indicator of the traffic type as this feature often is very high up on the tree, and thus is a very informative feature. The other features seem to be less consistent with their placement in the trees and more reflective of the data in question, for example, NUM of ACKs (number of ACK packets) in the TCP tree only. Moreover, in the UDP tree only 6 features out of 9 are employed (number of bytes in backward direction, number of ACK packets and protocol features are unused). On the other hand, in the TCP tree 7 of our 9 features are employed (duration and number of packets in the forward direction are unused). Thus, we can conclude that the feature vector appears to be robust enough to handle both TCP and UDP based Skype traffic, and still provide accurate classification.

7. Conclusions and Future Work

In this work, we have employed C4.5, and AdaBoost, both supervised learning algorithms for classifying Skype direct UDP, and Skype direct TCP communications from a given traffic file. To do so, the UCIS department at Dalhousie University provided us with a data set that had both UDP and TCP labeled traffic for Skype. We then converted these traces to traffic flows and tested the aforementioned learning algorithms using a feature vector to represent these flows.

Results so far show that both approaches perform with a very high detection rate and a low false positive rate when the feature set is employed. In this case, we have the worst case performance of 98% detection rate with 6% false positive rate achieved for UDP based Skype detection and the worst case performance of 94% detection rate with 4% false positive rate achieved for TCP based Skype classification by the C4.5 classifier. These results also indicate that the features selected to represent the traffic seem to be sufficient as well.

In summary, in this work, we have shown that it is possible to detect Skype Direct UDP and Direct TCP traffic without using features such as IP addresses and source/destination ports or any payload information. Indeed more analysis on the training data sets and labeling techniques need to be performed, as the results given here are preliminary but promising. Future work will test our classification system on more data sets captured on different networks, as well as with more varied Skype traffic including Skype tunneling traffic. In addition, future work will look at examining the accuracy of the labeled data set provided to us, to ensure that the classifier is being trained solely on Skype and not false positives of Packet Shaper style tools.

Finally, it might be interesting to examine the inner workings of Skype Super Node relay calls. In particular, looking at the call formats observed in other Skype sessions, and seeing if machine learning is appropriate or if simpler observed rules would be more fitting. While Skype direct VoIP communications can be successfully identified by our system, this does not mean that all Skype traffic can be identified, since there is still the matter of Skype Super Nodes, which can relay VoIP calls between peers, which is not studied in this work.

```
Src Byte Rate <= 0: 0 (338.0/14.0)
Src Byte Rate > 0
|   Maximum Pkt Size <= 408
|   |   Minimum Pkt Size <= 22: 1 (463.0/4.0)
|   |   Minimum Pkt Size > 22
|   |   |   Protocol <= 6
|   |   |   |   Minimum Pkt Size <= 47
|   |   |   |   |   Num of Acks <= 2
|   |   |   |   |   |   Src Byte Rate <= 38: 0 (2.0)
|   |   |   |   |   |   Src Byte Rate > 38: 1 (2.0)
|   |   |   |   |   Num of Acks > 2: 1 (5.0)
|   |   |   |   Minimum Pkt Size > 47
|   |   |   |   |   Num of Acks <= 1
|   |   |   |   |   |   Src Byte Rate <= 102.247879: 0 (2.0)
|   |   |   |   |   |   Src Byte Rate > 102.247879: 1 (4.0)
|   |   |   |   |   Num of Acks > 1: 0 (14.0)
|   |   |   Protocol > 6: 0 (17.0)
|   Maximum Pkt Size > 408
|   |   Src Num of Pkts <= 0.646439
|   |   |   Maximum Pkt Size <= 1030: 1 (12.0)
|   |   |   Maximum Pkt Size > 1030
|   |   |   |   Num of Acks <= 202
|   |   |   |   |   Dst Byte Rate <= 60.372803
|   |   |   |   |   |   Num of Acks <= 23: 0 (11.0/2.0)
|   |   |   |   |   |   Num of Acks > 23: 1 (4.0)
|   |   |   |   |   Dst Byte Rate > 60.372803: 0 (15.0)
|   |   |   |   Num of Acks > 202: 1 (4.0)
|   |   Src Num of Pkts > 0.646439: 0 (107.0/2.0)
```

Figure 1: TCP decision tree

```
Minimum Pkt Size <= 22
|   Maximum Pkt Size <= 17: 0 (26.0)
|   Maximum Pkt Size > 17
|   |   Maximum Pkt Size <= 407: 1 (408.0)
|   |   Maximum Pkt Size > 407
|   |   |   Maximum Pkt Size <= 901: 1 (14.0)
|   |   |   Maximum Pkt Size > 901
|   |   |   |   Minimum Pkt Size <= 19
|   |   |   |   |   Maximum Pkt Size <= 1272: 1 (3.0/1.0)
|   |   |   |   |   Maximum Pkt Size > 1272: 0 (5.0)
|   |   |   |   Minimum Pkt Size > 19: 1 (2.0)
Minimum Pkt Size > 22
|   Maximum Pkt Size <= 182
|   |   Duration <= 0.021418
|   |   |   Dst Num of Pkts <= 0.493594: 0 (54.0/1.0)
|   |   |   Dst Num of Pkts > 0.493594
|   |   |   |   Src Num of Pkts <= 0.493594: 0 (42.0/1.0)
|   |   |   |   Src Num of Pkts > 0.493594
|   |   |   |   |   Maximum Pkt Size <= 76: 1 (15.0)
|   |   |   |   |   Maximum Pkt Size > 76
|   |   |   |   |   |   Maximum Pkt Size <= 96: 0 (82.0)
|   |   |   |   |   |   Maximum Pkt Size > 96
|   |   |   |   |   |   |   Maximum Pkt Size <= 127
|   |   |   |   |   |   |   |   Dst Byte Rate <= 28: 0 (3.0)
|   |   |   |   |   |   |   |   Dst Byte Rate > 28: 1 (12.0)
|   |   |   |   |   |   |   Maximum Pkt Size > 127: 0 (35.0)
|   |   Duration > 0.021418
|   |   |   Maximum Pkt Size <= 126
|   |   |   |   Maximum Pkt Size <= 101
|   |   |   |   |   Maximum Pkt Size <= 83
|   |   |   |   |   |   Minimum Pkt Size <= 23: 0 (2.0)
|   |   |   |   |   |   Minimum Pkt Size > 23: 1 (21.0/4.0)
|   |   |   |   |   Maximum Pkt Size > 83: 0 (20.0)
|   |   |   |   Maximum Pkt Size > 101: 1 (26.0)
|   |   |   Maximum Pkt Size > 126: 0 (12.0)
|   Maximum Pkt Size > 182
|   |   Minimum Pkt Size <= 57
|   |   |   Duration <= 6.157436: 1 (180.0)
|   |   |   Duration > 6.157436
|   |   |   |   Minimum Pkt Size <= 29: 0 (6.0/1.0)
|   |   |   |   Minimum Pkt Size > 29: 1 (7.0)
|   |   Minimum Pkt Size > 57
|   |   |   Maximum Pkt Size <= 260: 1 (2.0)
|   |   |   Maximum Pkt Size > 260: 0 (23.0/1.0)
```

Figure 2: UDP decision tree

Acknowledgements

This is supported in part by MITACS, CFI and NSERC Discovery grants. This work is conducted as part of the NIMS project at http://www.cs.dal.ca/projectx/.

References

[1] SIP, http://www.cs.columbia.edu/sip/, last accessed October 2006.

[2] H.323, http://www.iec.org/online/tutorials/h323/, last accessed October 2006.

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

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

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

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

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

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

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

[10] Baset S. A., Schulzrinne, H., "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol", Proceedings of the IEEE INFOCOM, 2006.

[11] Garfinkel, Simon, "VoIP and Skype Security", 2005, http://www.simson.net/ref/2005/OSI_Skype6.pdf.

[12] Berson T., "Skype Security Evaluations", 2005, http://www.skype.com/security/files/2005-031%20security%20evaluation.pdf

[13] Lisha G., Junzhou L., "Performance Analysis of a P2P-Based VoIP Software", Proceedings of the IEEE AICT/ICIW, 2006.

[14] Biondi P., Desclaux F., "Silver Needle in the Skype", BlackHat Europe, 2006, http://www.secdev.org/conf/skype_BHEU06.handout.pdf

[15] Packeteer, http://www.packeteer.com/products/packetshaper/, last accessed October 2006.

[16] RTGuardian, http://www.facetime.com/productservices/rtguardian.aspx, last accessed October 2006.

[17] SonicWall IPS, http://www.sonicwall.com/products/gav_ips_spyware.html, last accessed October 2006.

[18] NetSpective, http://www.verso.com/enterprise/netspective/, last accessed October 2006.

[19] Cisco IOS 12.4, http://www.cisco.com/univercd/cc/td/doc/product/software/, last accessed October 2006.

[20] Surf Control, http://www.surfcontrol.com/Default.aspx?id=374&mid=3, last accessed October 2006.

[21] WEKA Software, http://www.cs.waikato.ac.nz/ml/weka/, last accessed June 2006.

[22] Alpaydin, Ethem, Introduction to Machine Learning, MIT Press, October 2004.

[23] Williams,N., Zander S., Armitage G.,"Evaluating Machine Learning Algorithms for Automated Network Application Identification", CAIA Technical Report 060410B, 2005.