Two-Phased Method for Identifying SSH Encrypted Application Flows

Matti Hirvonen
VTT Technical Research Centre of Finland
Kaitovayla 1, Oulu, Finland
Email: matti.hirvonen@vtt.fi

Mirko Sailio
VTT Technical Research Centre of Finland
Kaitovayla 1, Oulu, Finland
Email: mirko.sailio@vtt.fi

***Abstract*—The use of application-layer tunnels has become more popular nowadays. By using encrypted tunnels for prohibited application such as peer-to-peer file sharing it is easy to gain access to restricted networks. Application-layer tunnels provide a possibility to bypass network defenses which is even more useful for malicious users trying to avoid detection. The accurate identification of application flows in encrypted tunnels is important for the network security and management purposes.**

Traditional network traffic classification methods based on port numbers or pattern-matching mechanisms are practically useless in identifying application flows inside an encrypted tunnel, therefore another approach is needed. In this paper, we propose a two-phased method for classifying SSH tunneled application flows in real time. The classification is based on the statistical features of the network flows. The first classification phase identifies the SSH connection while the second classification phase detects the tunneled application. A simple K-Means clustering algorithm is utilized in classification. We evaluated our method using manually generated packet traces. The results were promising; over 94% of all flow samples were classified correctly, while untrained application flow samples were detected as unknown at high precision.

Keywords: Traffic monitoring, K-means, SSH analysis

I. Introduction

Accurate network traffic identification is crucial for network management and monitoring purposes. It is especially needed in detecting security breaches and prohibited application flows. From a Quality of Service (QoS) perspective, it is important to be able to assign an approriate QoS level to a network flow.

At the same time, the use of encryption to protect user data has become more popular in today's Internet. This reduces the network operators' capability to secure their network from unknown threats, as encapsulating the flows inside encrypted tunnels can be used to bypass firewalls and Intrusion Detection Systems (IDS). Without an efficient application-layer tunnel detection, encapsulated prohibited application flows cannot be detected.

Security policy can restrict what protocols and services are allowed in the network. The policies' purpose is to limit the risk by keeping up network availability, restricting network access only to legitimate users and securing the integrity of the data in the network. Moreover, it restricts the used protocols and applications which give no additional value to the network owner. This aims at limiting the amount of vulnerable interfaces and services available for outside attackers.

The most common methods for identifying network flows are based on TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) port numbers or IP (Internet Protocol) packet's payload. Unfortunately, these methods are inadequate for identifying SSH[1] (Secure Shell) tunneled applications. First, because the available port number belongs to the SSH application. Secondly the packet payload is encrypted. The encryption makes the Deep Packet Inspection (DPI) process computationally impractically expensive. DPI has also problematic privacy issues concerning the user data.

This paper presents a two-phased method for identifying application flows inside SSH tunnels. The first classification phase identifies the SSH connection early at the TCP connection and the second classification phase recognizes the hidden application flow inside the SSH tunnel later in the flows life cycle. The results of this work clearly show that SSH tunneled flows can be identified accurately using only the statistical features of the flows.

The rest of the paper is organized as follows. Section II introduces relevant state-of-the-art studies in this field. The classification methodology is presented in Section III, while the behavior of the tunneled applications are investigated in Section IV. Section V illustrates the classification results, Section VI holds discussion on the results and Section VII concludes this work.

II. Related Work

Studies related to application identification from encrypted tunnels have been presented in recent publications. Many researchers have used packet arrival times and packet size as distinguishing features [3] [7] [12].

In [12] Wright *et al.* describe a Markov-model approach for identifying tunneled applications utilizing the sequence of tuples consisting of packet size and arrival time for each packet in the connection. They show that encrypted tunnel with a single application protocol gives enough information about the flows in the tunnel to allow to accurately track their number.

Also Dusi *et al.* [3] and Horton *et al.* [7] utilize information on packet sizes and packet arrival times for the classification. Study presented by Horton *et al.* [7] aims at detecting network

[1]SSH is a network procotol for secure remote access. It is defined in RFC 4250-4256

978-1-4577-9538-2/11/$26.00 ©2011 IEEE

attacks and uses traffic analysis on SSH protocol to classify different sessions and detect possible policy violations. In [3], Dusi *et al.* describe statistical approach into detecting tunneled flows by using packet size, arrival interval and arrival order as defining characteristics of the flow. Their work present a statistical fingerprinting method which is capable of identifying tunneled application protocols on top of SSH connections. Their method detects SSH tunneled application flows with more than 87% accuracy. SSH tunneled applications consist of email, chat and peer-to-peer applications.

Bernaille *et. al.* [2] described an application detection from a SSL (Secure Sockets Layer) encrypted flow. They identified tunneled applications utilizing only the first packets of the connection. Therefore their method cannot provide continuous classification. Bernaille *et. al.* [2] stated that they are able to classify applications inside an SSL connection with more than 85% accuracy.

Maiolini *et. al.* [8] describe a real time K-means based identification algorithm for SSH encrypted application flows. They utilized arrival times, sizes and directions of packets in the classification. Study stated that they gained accuracy up to 99.88% in identifying HTTP over SSH by analyzing only the third and the fourth packet after the end of SSH initiation phase. They focused on SSH services such as SCP, SFTP and HTTP over SSH. They also detected a decrease in accuracy when adding additional applications for detection. The present paper also utilizes K-means clustering algorithm, but the number of target applications is greater than in [8], including widely used BitTorrent protocol. It is noticeable that, with a greater number of tunneled application, the classification accuracy remains high.

The method described in this study is able to classify tunneled network flows in real time. Many of the recent studies [3], [7], [8], [12] use packet arrival times as a feature. This study does not use arrival times, because the network conditions can have a huge effect on packet inter-arrival times.

III. Classification Methodology

This section describes the two-phased classification method for identifying tunneled application flows[2] inside an SSH connection.

The first classification is executed early at the connection, when the first four packets of the connection which contain any payload are captured. If the result of the first classification phase indicates that the current flow is SSH, the classifier instructs the second classification phase to look for the tunneled application. The second classification phase is done later, when enough packets have arrived, and is repeated periodically every time when a required amount of packets of the particular flow has been collected. This study focuses on the second classification phase.

The identification system consists of two distinct stages, training and classification. These stages are depicted in Figure 1 and described shortly in Sections III-A and III-B. This study and our previous study [6] utilize similar stages to the one presented by Bernaille *et. al.* [1]. Our previous work enhanced Bernaille's study [1] to classify TCP flows in real time with a two-phased method. This work extends our two-phased method further in identifying SSH tunneled flows. More detailed information about the stages can be found from our previous studies, [5] and [6].

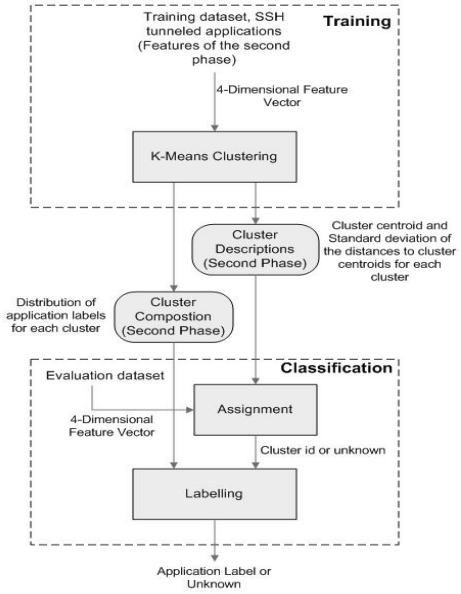

Fig. 1. Training and classification stages.

A. Training Stage

The training stage applies a K-means clustering method [9] to a set of training data in order to group network flows into distinct clusters based on similar behavior. K-means clustering takes the training dataset as input and produces two separate cluster files; cluster descriptions and cluster compositions. The cluster description file includes the positions of the clusters in the feature space and corresponding measures of how wide are the clusters. The cluster composition file contains information about how the trained applications are distributed into the previously mentioned clusters [5] [6].

The K-Means algorithm requires information on number of clusters and the initial cluster centroids before it can be initiated. In order to select the best possible number of clusters and initial cluster centroids, the quality of clustering is calculated using two different clustering quality measures: Normalized Mutual Information (NMI) and purity [9]. Due to lack of space quality measures could not be discussed in this paper. According to the quality measures the best trade-off between the quality of clustering and the number of cluster was found when the number of cluster is 50. Initial cluster centroids were selected based on empirical experience.

[2]The term *flow* refers to a bi-directional network flow which is defined by a 5-tuple consisting of source IP address, source port, destination IP address, destination port and transport layer protocol number

B. Classification Stage

The classification procedure utilizes the output of the training stage and contains two substages; assignment and labeling. In the assignment stage, a flow is assigned to a cluster or labelled as unknown, if it does not belong to any of the cluster's coverage areas. The assignment stage also utilizes an user defined threshold value, which can be used to change cluster coverage areas. A larger threshold value means larger clusters and more flows will be labeled with a certain application name. When selecting the best possible threshold value, trade-off between the classification accuracy and detection of unknown traffic exists. In our case, the chosen threshold value is 7. The effect of the threshold value on the performance of the classifier is no further discussed in this paper due to lack of space. The labeling stage labels a flow with an application name. The rule used in this study was to label a flow with the most dominating application in given cluster.

IV. Behavior Analysis of Application Traffic

At the beginning of this section the datasets are introduced, and the way they were collected is described. The discussion continues with the introduction of tunneled applications and proceeds to the analysis of flow features.

A. Datasets

Packet traces play a key role in the learning and evaluation process. Collected packet traces are divided into two datasets i.e, training and evaluation. The training dataset is supposed to be as extensive as possible. It should cover all possible behavior types of the target applications.

All the packet traces used in this study were generated manually in laboratory environment. It is because there were no publicly available traces available with the ground truth information. To verify the ground truths of the application flows, each application trace was generated separately one at a time in an isolated environment.

The features for the first phase was collected from the first packets of the connection, while the features for the second phase was extracted from a packet period of 800 packets starting after 200th packet. Then the flow is assumed to be in its steady state. This means that an initiation of the flow is done and the actual data transfer has begun. Flow features collected from the packet period of 800 packets are called as a flow sample.

This study focuses on classifying five different applications, HTTP (HyperText Transfer Protocol), FTP (File Transfer Protocol), BitTorrent, SHOUTCast (audio streaming), and IRC (Internet Relay Chat). The dataset contains different behavior types of these previously mentioned applications, for example HTTP web browsing and file transfer, FTP filetransfer and other FTP operations. The next section discusses how those behavior types appear for the classifier.

B. Flow Features

Selection of flow features can have a huge effect on the performance of the classifier. The basic idea is to find features which can be used to distinguish application flows from others.

Previous study [6] has shown that SSH flows can be classified accurately using only the sizes and the directions of the first four packets of the connection as flow features for classification. It is because the first packets capture the applications negotiation phase which is usually a pre-defined sequence of messages and is usually different among different applications. Packets that do not contain any payload are excluded (TCP-control packets, like SYN or ACK). These features cannot be used to classify SSH tunneled application, because the application flow has not started yet. The focus of this study is on the classification of SSH tunneled applications for more detailed explanation of the first classification phase can be found from [5] and [6].

The selection of flow features was based on author's own studies utilizing several feature selection algorithms, including Correlation-based Feature Selection (CFS) algorithm [11]. Features which are dependent on packet inter-arrival times were excluded, because different network conditions can have a major effect to those. More discussion on feature selection and extraction is available at [5]. The WEKA software suite [11] was utilized in ranking the flow features. Formulas of the selected flow features are,

1) $Average\ Packet\ Size$
2) $100 * \frac{Total\ Packet\ Size\ (DL)}{Total\ Packet\ Size}$
3) $100 * log_{10}\frac{\#Packets\ with\ PUSH\ Flag\ (DL)}{\#Packets\ with\ PUSH\ Flag\ (UL)}$
4) $100 * log_{10}\frac{\#Packets\ with\ Payload\ (DL)}{\#Packets\ with\ Payload\ (UL)}$

, where DL indicates downlink direction, UL uplink direction of the packets.

Figures 2 and 3 illustrate how flow samples of target applications are distributed in a selected feature space. For example IRC, BitTorrent and audio streaming samples form tight clusters, whereas HTTP and FTP spread more widely around the feature spaces. A dot in a figure represents a sample of the target application flow.

Some applications also tend to mix with others in the selected feature space. For example, in upper-right part of the Figure 2, FTP and HTTP flow samples are distributed in a wide area. The same kind of mingling between FTP and HTTP samples can be found in the upper-right corner of the Figure 3. As mentioned earlier, applications have different behavior types which can be distinguished by the features. HTTP and FTP share a common behavior type i.e, downloading a file. This common behavior type explains the mixing in the upper right parts of the Figures 2 and 3. Other HTTP flow samples in the figures are web browsing traffic, which seems to be very diverse.

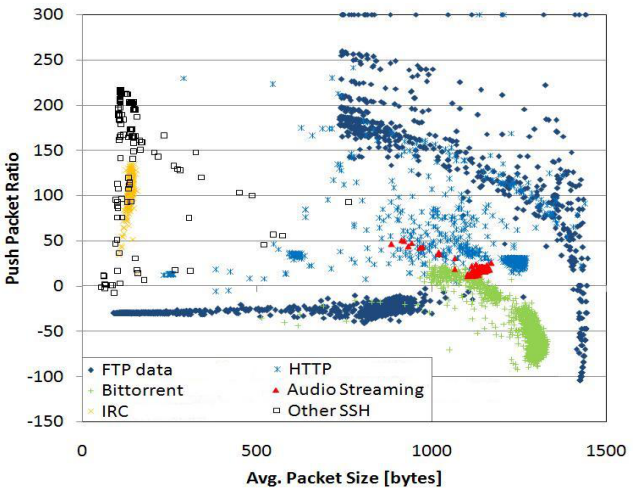

Fig. 2. Feature space part one; features 1 and 3.

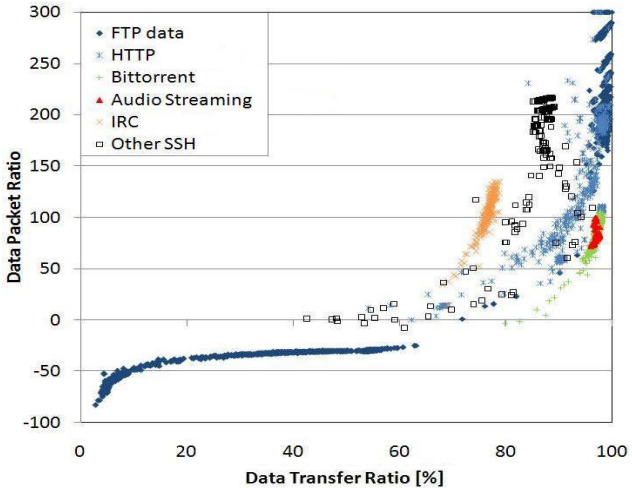

Fig. 3. Feature space part two; features 2 and 4.

In addition, FTP traffic contains other behavior types like, uploading to the server, file removal from the server, folder name changes, etc. These behavior types can be also distinguished from the Figures. FTP uploading appears approximately in the middle of the Figure 2, where the average packet size is between 800 and 1000 bytes. FTP uploading can be also found from the lower left corner in Figure 3. Other FTP operations tend to use small packets and the flow samples are located in the middle of the left side of the Figure 2. Figure 3 depicts the samples of other FTP operations in lower part of the figure, where data transfer ratio is around 50%.

Audio streaming traffic seems to have the most compact cluster of samples according to the Figures 2 and 3, and the behavior seems to be similar in Internet radios with different bit rates. According to these figures, The IRC traffic samples can be detected most easily among other applications. These samples do not mingle with other samples much. IRC traffic consists of small sized packets, which is one of the best discriminating features in case of classifying IRC flows. In a BitTorrent case, the combination of the Features 1 and 3 appears to be good for discriminating BitTorrent from others. BitTorrent flow samples seems to mix a bit with audio streaming and HTTP samples.

V. Experimental Results

This section discusses on the evaluation results of the presented classification system. At first, the evaluation metrics are shortly introduced and then the classification results of SSH tunneled applications are presented.

Typically the performance of a classification system is represented by metrics as, True Positives, False Positives, False Negatives and True Negatives. This study utilizes these widely known metrics in evaluation. The definitions of these metrics can be found e.g. in a paper by Nguyen *et. al.* [10].

A. Identification of SSH tunneled applications

The identification accuracy of SSH tunneled application flows was tested using an evaluation dataset. The evaluation dataset consists of flow samples of SSH tunneled applications. 200 flow samples were selected randomly for the training dataset per target application. Over 300 flow samples per application was left to evaluation dataset.

Other SSH flows were also generated in order to evaluate the classifiers ability to detect unknown traffic. Other SSH traffic includes reading man-pages, opening, creating and modifying files, as well as writing Linux commands in an SSH server. This traffic was not trained to the classifier.

Overall, the classifier identified 94.1% of evaluation flow samples correctly, while over 91% of the untrained traffic flows were classified as unknown. Table I illustrates the classification results of SSH tunneled application flows with a chosen threshold value. Bolded results indicate true positive values and other values represent false negatives. The header row of the Table I shows the list of true applications and the first column then reveals the list of classification results.

TABLE I
Confusion matrix, classification of SSH tunneled applications

		Real Application					
	Appl.	Audio	BT	FTP	HTTP	IRC	Other
Class. Result	Audio	**97.65**	2.03	0.00	0.00	0.00	0.00
	BT	2.35	**96.65**	0.09	1.87	0.00	0.00
	FTP	0.00	0.00	**90.84**	6.78	0.00	3.14
	HTTP	0.00	1.28	6.45	**89.72**	1.49	2.88
	IRC	0.00	0.00	0.00	0.00	**98.52**	2.62
	Unknown	0.00	0.04	2.62	1.64	0.00	**91.36**

The classification accuracies are very high in case of Audio, BitTorrent and IRC applications. All those applications are accurately distinguished from other applications, as the results of the clustering and the Figures about the feature space show.

The classification results also confirm that HTTP and FTP traffic share similar type of behavior, since those application

flows mix up with each other. 6.78% of HTTP flow samples are classified as FTP and approximately 6.45% of FTP flow samples are identified as HTTP. Also audio flows were classified as BitTorrent and vice versa in some cases. This behaviour is due to the fact that these pairs of applications located very close to each other in the feature space and appear in same clusters in the training set. The results show that mingling of HTTP file transfer and FTP download is largely affecting to the classification results. In overall, the evaluation of the method produce low procentage of false negatives.

VI. Discussion

Our method is capable of identifying the flow-behavior of multiple protocols within encrypted tunnels. The overall 94% detection rate and the low false positive rates are promising. This work shows that the type of an application can be detected from an SSH encrypted flow using only the statistical features of the application flows.

The results show also that applications sharing similar behaviors are often easily mixed with each other, as evident with FTP and HTTP. To prevent this it could be more convenient to form classes based on the type of behavior. For example there could be a class called file transfer which would include file transfer behaviors of different applications. That kind of a class partitioning in a network traffic classifier would be suitable for both quality of service management and information security monitoring purposes.

It can be said that the behavioral based analysis suited well to identifying applications inside encrypted tunnels. Advantages compared to deep packet inspection systems are that the classification process is light-weighted, as it does not require much processing power or memory, and there are less potential for privacy violation as the packet payload is not needed. One major limitation of the behavioural based classifiers is that the features like packet sizes can be altered. Our method is also limited in assuming that inside a SSH tunnel there is only one application. If the tunnel contains traffic of multiple types of applications simultaneously the performance of the classifier would probably deteriorate. How much this would effect the performance of the classifier requires further study.

Our method requires 1000 packets before the first classification of a tunneled application. With a packet size of 1.5 kbytes almost 1.5 Mbytes can be transferred without identifying the application. Especially in the security context it is important to identify threats quickly. This can be achieved by using smaller flow sample sizes.

The online performance of the classifier is dependent on the training dataset in other words how widely different behavior types of the applications have been captured and taught to the classifier. Therefore a more extensive training dataset would provide more reliable online classification.

Other plans for the future are to generate more extensive dataset and extend this analysis to other encrypted tunnels.

VII. Conclusion

In this paper, we propose a statistical method for identifying application types within SSH-encrypted tunnels. The classification is performed in two-phased manner, where the first phase detects the SSH connection and the second phase identifies the encrypted application flow. Our method is able to provide real time flow classification for the lifetime of the flow. The method provided promising results with 94% of applications classified correctly gaining low false positive percentage.

This work shows that the type of application inside the SSH tunnel can be recognized accurately by utilizing only the statistical features of the application flows. This method enables the detection of prohibited or low priority application flows inside encrypted tunnels. This would be beneficial in QoS management and network use control purposes.

It can be said that the behavioral based analysis suited well to identifying applications inside encrypted tunnels. Advantages compared to deep packet inspection systems are that the classification process is light-weighted, because it does not require much processing power or large memory storage, and there are no serious privacy concerns because access to the packet payload is not necessary.

Acknowledgment

This work has been performed in MOVERTI project, which is co-funded by Tekes (Finnish Funding Agency for Technology and Innovation). This work was also supported by the IST Games@Large project, partially funded by the European Commission.

References

[1] L. Bernaille, R. Teixeira, K. Salamatian. Early application identification. In *Proceedings of the 2006 ACM CoNEXT conference (CoNEXT '06)*, December 04-07, 2006, Lisboa, Portugal, [doi¿10.1145/1368436.1368445].

[2] L. Bernaille and R. Teixeira. Early recognition of encrypted applications. In *PAM'07: Proceedings of the 8th international conference on Passive and active network measurement*, pages 165–175, Berlin, Heidelberg, 2007. Springer-Verlag.

[3] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli. Detection of encrypted tunnels across network boundaries. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 1738 –1744, 19-23 2008.

[4] V. Foroushani, F. Adibnia, and E. Hojati. Intrusion detection in encrypted accesses with ssh protocol to network public servers. In *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pages 314 –318, 13-15 2008.

[5] M. Hirvonen. *Two-Phased Network Traffic Classification Method for Quality of Service Management*. University of Oulu, Finland, 2009. Master's Thesis.

[6] M. Hirvonen and J.-P. Laulajainen. Two-phased network traffic classification method for Quality of Service management. In *Consumer Electronics, 2009. ISCE '09. IEEE 13th International Symposium on*, pages 962 –966, 25-28 2009.

[7] J. Horton and R. Safavi-Naini. Detecting policy violations through traffic analysis. In *ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference*, pages 109–120, Washington, DC, USA, 2006. IEEE Computer Society.

[8] G. Maiolini, A. Baiocchi, A. Iacovazzi, and A. Rizzi. Real time identification of ssh encrypted application flows by using cluster analysis techniques. In *NETWORKING '09: Proceedings of the 8th International IFIP-TC 6 Networking Conference*, pages 182–194, Berlin, Heidelberg, 2009. Springer-Verlag.

[9] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 1 edition, July 2008.

[10] T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys Tutorials, IEEE*, 10(4):56 –76, fourth 2008.

[11] T. U. of Waikato. Weka: Data mining software in java. Last Accessed January 2010, http://www.cs.waikato.ac.nz/ml/weka/.

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