On the Fly Encoded Application Flows Recognition by Relying on Statistical Features of IP Traffic

Gianluca Maiolini[1], Andrea Baiocchi[2], Antonello Rizzi[2], Sara Ferri[1] and Letizia Gabbrielli[1]

[1] AMTEC SpA, Loc. San Martino,
Piancastagnaio, SI, Italy,
{gianluca.maiolini, sara.ferri, letizia.gabbrielli}@elsagdatamat.com

[2] INFOCOM Dept., University of Roma "Sapienza"
Rome, Italy,
{andrea.baiocchi, antonello.rizzi}@uniroma1.it

Abstract. The secure collaborative judicial workspace (SCJW) has to allow the actors to use a number of communication and scheduling instruments for managing and storing any kind of documentation, video and audio recordings, evidence, among different Judicial offices of different countries. In this scenario is very important to identify encoded application delivering those application services to guarantee secure communication, but at the same time it is important to not compromise privacy of information exchanged. In this paper we aim at identifying application flows encoded within SSH tunnels by relying on statistical feature of IP packets. This will enable SCJW network administrator to identify un-trusted applications without analyze traffic contents.

Keywords: Traffic analysis, statistical traffic classification, SSH, cluster analysis, k-means.

1 Introduction

One of the most critical aspects every government should consider in the context of such a modernization is the field of justice. The most prominent issues is guaranteeing that any information flowing within judicial information systems is treated in a secure manner. In a cross border judicial cooperation during investigations, the information flows between different actors, different systems and at different levels. These information are very sensitive, they should be protected from unauthorized access and should be accessed only by specific people according to their role in the judicial process. Moreover document transfer from one country to another country must comply with the requirements of non repudiation. In a generic request of cross-border judicial cooperation one independent platform will support the country requesting judicial cooperation and other platform will support the country providing judicial cooperation. The secure collaborative judicial workspace (SCJW) has to allow the actors to use a number of communication and scheduling instruments for managing and storing any kind of documentation, video recordings, audio recordings, evidences etc, among different Judicial offices of different countries. So, the SCJW has to deal with:

- the request and exchange of critical evidence documentation,
- the request for remote interrogations via videoconference,
- the request for specific actions, such as phone interception in another country,
- the request for warrant of arrest.

In this context is basic a fast traffic classification means according to the services data is generated by. Secure Shell or SSH is a network protocol that allows data to be exchanged using a secure channel between two networked devices. It is often used to login to a remote computer but it is also applied for tunneling, file transfer and forwarding arbitrary TCP ports over a secure channel between a local and a remote computer. What makes the detection of this protocol interesting is that its traffic is encrypted. Thus any payload analysis based classification method is irrelevant since the payload is encrypted. Actually DPI technology cannot recognize application delivered within SSH flows.

The objective of our work is to develop a real time system to recognize and classify SSH flows by analyzing statistical features of first IP packets belonging to a SSH connection, such as directions and lengths. This enables us to identify service applications without compromise privacy of contents exchanged by users during network communications. By recognition we mean identifying which flows belong to SSH protocol as opposed to other application level protocols. By classification we mean to identify the kind of service carried within each SSH connection, such as SCP, SFTP and HTTP over SSH. Experiments show that our approach permits us to achieve great recognition accuracy up to 99.2% for SSH identification and, once SSH has been identified, applications in those SSH tunnels are classified with accuracy up to 99.8%.

2 Related Work

Different approaches to traffic classification have been developed, using information available at IP layer such as inter-arrival times, bytes transferred, packet size. Some proposals [4][5] need also semantically complete TCP flows as input.

In [1], Karagiannis et al. developed a heuristic that uses social, functional and application level behaviours of a host to identify all traffic flows originating from it. This approach, although really innovative, is tailored onto a specific source host.

Salgarelli et al. [2] used only size and inter-arrival time of first *n* packets to create a statistical descriptor (a Fingerprint) of an application layer protocol: this fingerprint is then used to measure the similarity of a certain flow to the corresponding protocol.

The Hidden Markov Models (HMM) theory is used in [3]: packets size and inter-arrival time are used to build a model describing a certain protocol. The results of the training phase is a HMM model describing the behaviour of each protocol. Even though this approach can classify distinct encrypted applications, its performance on

SSH is (76% detection rate and 8% false negative) is not as good as well known application traffic such as WWW and instant messaging.

Moore et al. [4] used a supervised machine learning algorithm called Naive Bayes (and its generalization, Kernel Estimation) on a wide set of characteristics (tens or hundreds), as flow duration, packets inter-arrival time and payload size and their statistics (mean, variance...). Moreover, they use a filtering technique to identify the best characteristics to be used with the mentioned methods.

A number of works [5][6][7] rely on unsupervised learning techniques. McGregor et al. [5] explore the possibility to use cluster analysis to group flows using transport layer attributes, but they do not evaluate the accuracy of the classification. Zander et al. [6] extend this work using another Expectation Maximization (EM) algorithm named Autoclass. They also analyze the best set of attributes to use. Both these works only test Bayesian clustering technique trained by an EM algorithm, which has a slow learning time.

Bernaille et al. [7] use faster clustering algorithms representing data in different spaces: K-means and Gaussian Mixture Models (GMM) for euclidean space and Spectral clustering in HMM based space. The only features they use are packet size and packet direction: they demonstrate the effectiveness of these algorithms even using a small number of packets (e.g. the first four of a TCP connection).

Alshammari et Al [8], work attempted to classify/identify applications services running over SSH. They have shown the utility of two supervised learning algorithms AdaBoost and RIPPER for classifying SSH traffic without using features such as payload, IP addresses and source/destination ports. Results indicate that a detection rate of 99% and a false positive rate of 0.7% can be achieved using RIPPER. Moreover, promising preliminary results were obtained when RIPPER was employed to identify which service was running over SSH. They can recognize applications inside SSH flows such SCP and SFTP with accuracy up to 99.8% but they have performed off-line analysis on complete traces. We aim at classifying applications inside SSH flows in real time mode just analyzing the firsts 4 packets after SSH negotiation. We rely on K-means cluster analysis machines algorithm.

3 Problem Statement

In this paper, we focus on the classification of IP flows generated from network applications communicating through TCP protocols. Our objective is to recognize SSH flows out of other applications such as HTTP, FTP, POP3, etc. and, once that is accomplished, to identify which service is actually carried within the encrypted SSH tunnel. Then, we first need to define exactly what we mean for TCP flow.

Definition: A flow F is the bi-directional, ordered sequence of IP packets exchanged during a TCP connection.

Within a TCP connection, application level data are delivered as well as control packets, such as those related to three way-handshake (RFC-793) and TCP ACK packets. So, TCP flow will be composed by packets from SYN (PK_0) to FIN (PK_{N-1}). Each flow could be seen as a sequence of (PK_0, …., PK_{N-1}), where PK_j represents the j-th IP packet exchanged during TCP connection. Since we aim at classifying application flows relying on statistical features of IP packets, such as length, direction, we will characterize each TCP flow F as an ordered sequence of N-tuples $(d_j, l_j. t_j)$, with $0 \leq j \leq N-1$, where:

- $d_j \in [0,1]$ where 1 encodes the direction detected for SYN packet and 0 the opposite direction;
- l_j length of IP $PK_{j,}$ in bytes;

The packet length ranges between a minimum and a maximum. The latter is the MTU (Maximum Transmission Unit) of the interfaces crossed by TCP connections packets. In all experiments we found out MTU=1500 bytes has never been exceeded, which is just the largest allowed MTU of most Ethernet LANs and hence most of the Internet [11]. As for the minimum length, it corresponds to those carrying a TCP ACK and is denoted as l_{ACK} in the following. It is the smallest length detectable for a TCP packet as we tested during our experiments and as RFC 793 refers, typical values ranging between 40 and 56 bytes, depending on options in the TCP and IP headers.

4 Dataset Creation

Given our aim as stated in the introduction, we assume a trained machine learning approach, exploiting cluster analysis. To that end, we need both a test and a train data set. A data set for our purposes is composed of a collection of flows in the sense defined in Section III along with metadata per flow, reporting the *known* application layer protocol the flow belongs to.

Knowing the application protocol each flow belongs to is needed to reliably train our algorithm. Since publicly available traces have payloads stripped off (for obvious privacy reason, e.g. CAIDA traces) and classification results cannot be checked reliably, we resorted to artificial traffic carefully generated by exploiting network premises at the University campus, the Elsag Datamat site and a private home. This way we encompass three major kinds of Internet access points: institutional, business and domestic. The controlled traffic generation is a must specifically for collecting SSH traces whose service content is known, i.e. to further label each SSH flow with a metadata reading which service it is carrying among SCP, SFTP and HTTP.

4.1 Data Collection

Our data collection approach is to simulate possible network scenarios using one or more computers to capture the resulting traffic. In order to have realistic traces and technology independent implementations of SSH (version 2) protocol, we used computers with heterogeneous operative systems, namely Linux and Windows. We simulate SSH connections by connecting three client computers deployed in three different LAN to one server. As shown in figure 1, client LANs and SSH server have been connected to the Internet by using different geographic links. We run the following SSH services: SCP, SFTP and HTTP over SSH. SCP and SFTP are transfer file services natively available on OpenSSH [10]. In particular we downloaded/uploaded files from clients to server using both SCP and SFTP protocols collecting eight thousands flows. HTTP over SSH traces have been collected downloading web pages through SSH tunnels (one SSH tunnel for each HTTP session). We get four thousands of flows.

SSH connections can tunnel several TCP flows at the same time: we are working in the case where each flow is assigned by SSH a separated channel, each with specific SSH identifier. Finally we will consider flows without SSH compression feature.

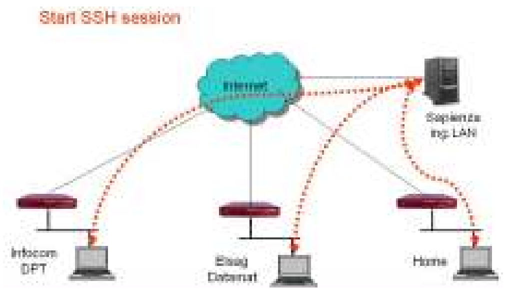

Fig. 1: Platform used to generate SSH traffic: SSH server is inside the University campus network; clients are at University, Elsag Datamat and a private home premise, respectively.

4.2 Data Set Creation: Pre-processing of Traces

In order to create data sets we pre-processed collected traffic traces. In particular we think that removing packets related to TCP control messages from each flow *F* can help us highlighting the differences among various applications. Therefore we remove from each flow *F* packets related to:

- Three-way handshake of TCP: $PK_{0=SYN}$, $PK_{1=SYN\text{-}ACK}$, $PK_{2=ACK}$;
- TCP ACK packets, i.e. those packets carrying only a TCP level ACK and no payload data;
- Retransmitted packets.

According to TCP protocol (RFC 793) the third packet ($PK_{2=ACK}$) of each TCP connection flow *F* carries an ACK. In order to remove ACK packets and TCP header length at the same time, we detect $PK_{ACK} = <d_{ACK}, l_{ACK}>$ of each session, where:

- d_{ACK} is 1, because ACK direction in three way handshake is always consistent with that of SYN packet;
- l_{ACK}, is the length of packet containing TCP ACK;

We aim at identifying application within SSH tunnels. Then, we further process SSH flows by removing packets related to the SSH initial handshake (see Figure 2), these packets are easily recognizable thanks to a specific pattern in terms of length and direction of packets exchanged when a new SSH channel is open. We consider the following services inside encrypted SSH tunnels: SCP, SFTP and HTTP over SSH.

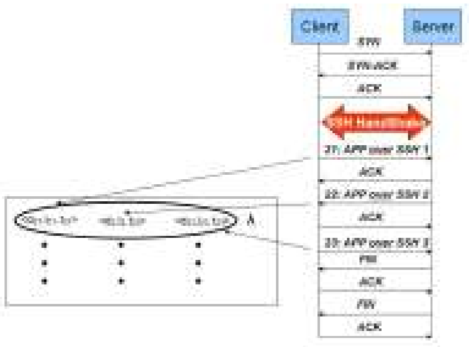

Fig. 2: Pre-processing of SSH flows

5 Classification Method

In this section some details about the adopted classification system are exploited. Basically a classification problem can be defined as follows. Let $P : X \rightarrow L$ be an unknown oriented process to be modeled, where X is the domain set and the codomain L is a label set, i.e. a set in which it is not possible (or misleading) to define an ordering function and hence any dissimilarity measure between its elements.

If P is a single value function, we will call it *classification function*. Let S_{tr} and S_{ts} be two sets of input-output pairs, namely the training set and the test set. We will call *instance* of a classification problem a given pair (S_{tr} , S_{ts}) with the constrain $S_{tr} \cap S_{ts} = \emptyset$. A classification system is a pair (M , TA_i), where TA is the training algorithm, i.e. the set of instructions responsible for generating, exclusively on the basis of S_{tr}, a particular instance $\overline{\mathrm{M}}$ of the classification model family M, such that the classification error of $\overline{\mathrm{M}}$ computed on S_{ts} will be minimized. The *generalization capability,* i.e. the capability to correctly classify any pattern belonging to the input space of the oriented process domain to be modeled, is for sure the most important desired feature of a classification system. From this point of view, the mean classification error on S_{ts} can be considered as an estimate of the expected behavior of the classifier over all the possible inputs. In the following, we describe a classification system trained by an unsupervised (clustering) procedure.

When dealing with patterns belonging to the R^n vectorial space we can adopt a distance measure, such as the Euclidean distance; moreover, in this case we can define the prototype of the cluster as the centroid (the mean vector) of all the patterns in the cluster, thanks to the algebraic structure defined in R^n. Consequently, the distance between a given pattern x_i and a cluster C_k can be easily defined as the Euclidean distance $d(x_i ; \mu_k)$ where μ_k is the centroid of the pattern belonging to C_k:

$$\mu_k = \frac{1}{m_k} \sum_{x_i \in C_k} x_i \ . \tag{1}$$

A direct way to synthesize a classification model on the basis of a training set S_{tr} consists in partitioning the patterns in the input space (discarding the class label information) by a clustering algorithm (in our case, by the K-means).
Successively, each cluster is labeled by the most frequent class among its patterns. Thus, a classification model is a set of labeled clusters (centroids); note that more than one cluster can be associated with the same label, i.e. a class can be represented by more than one cluster. Assuming to represent a floating point number with four bytes, the amount of memory needed to store a classification model is $K \cdot (4 \cdot n + 1)$ bytes, where n is the input space dimension and assuming to code class labels with one byte. An unlabeled pattern x is classified by determining the closest centroid μ_i (and thus the closest cluster C_i) and by labeling x with the same class label associated with C_i. It is important to underline that, since the initialization step of the K-Means is not deterministic, in order to compute a precise estimation of the performance of the classification model on the test set S_{ts}, the whole algorithm must be run several times, averaging the classification errors on S_{ts} yielded by the different classification models obtained in each run.

6 Experimental Results

By classifying plain SSH flows with our approach, we obtained results shown in table 1. We tried out processing all possible combination of packets up to ten packets after end of SSH negotiation (i.e. the initial common handshake phase, same in all SSH flows).

Table 1. Encoded SSH applications flows

1°	2°	3°	4°	5°	HTTP over SSH	scp	sftp
0	0	1	1	1	99.80%	98.93%	99.75%
0	0	1	1	0	99.88%	99.30%	99.05%

As shown in Table 1, we tested different patterns representations, increasing the considered number of packets for each flow in order to identify which one contains more information to emphasize difference among applications. As shown in table 1, the K-means based algorithm yields very interesting results in terms of identification of encoded applications. We can detect different applications with accuracy up to 99.8 for HTTP over SSH protocol, just analyzing third and fourth packets after SSH negotiation. We can notice that analyzing also the fifth packet does not improve significantly accuracy. Moreover, increasing the considered number of packets means introducing delay for real time recognition.

7 Conclusion

In this paper we present a model that could be useful to address the problem of traffic classification. To this end, we use only (poor) information available at network layer, namely packets size, directions and inter-arrival times. Our classification system based on cluster analysis can classify in real time encoded SSH traffic flows, overcoming actual limits of deep packet inspection. Our system does not compromise privacy of network users because to identify encoded SSH tunnel payload information are not inspected.

We are able to identify the nature of each SSH tunnel obtaining accuracy up to 99.88% in classifying HTTP over SSH just analyzing the third and fourth packet after the end of the SSH negotiation phase. The same encouraging results have been obtained by classifying SCP (up to 99.3) and SFTP (up to 99.05) applications. Further works should be performed in order to improve results for classification of download and upload flows for SCP and SFTP. Moreover, it will be necessary to investigate the applicability of the approach on wider application dataset.

Currently on-going work includes extension of the classification tool to more powerful classification algorithms, well beyond k-means; in this respect, k-means shall be regarded as a first use attempt, to verify the soundness of our approach, before proceeding to more complex yet reliable classification algorithms.

References

1. Karagiannis, K. Papagiannaki, M. Faloutsos, ”BLINC: Multilevel traffic classification in the dark”, Proc. of ACM SIGCOMM 2005, Philadelphia, PA, USA, August 2005.
2. M. Crotti, M. Dusi, F. Gringoli, L. Salgarelli, ”Traffic Classification through Simple Statistical Fingerprinting”, ACM SIGCOMM Computer Communication Review, Vol. 37, No. 1, pp. 5-16, Jan. 2007.
3. C. Wright, F. Monrose, G. Masson, ”On Inferring Application Protocol Behaviors in Encrypted Network Traffic”, Journal of Machine Learning Research (JMLR): Special issue on Machine Learning for Computer Security, volume 7, pp. 2745-2769, 2006.
4. A.W. Moore, D. Zuev, ”Internet traffic classification using Bayesian analysis techniques”, ACM SIGMETRICS 2005, Banff, Alberta, Canada, June 2005.
5. A. McGregor, M. Hall, P. Lorier, J. Brunskill, ”Flow clustering using machine learning techniques”, PAM 2004, Antibes Juan-les-Pins, France, April 2004.
6. S. Zander, T. Nguyen, G. Armitage, ”Automated traffic classification and application identification using machine learning”, LCN 2005, Sydney, Australia, November 2005.
7. L. Bernaille, R. Teixeira, and K. Salamatian, ”Early Application Identification”, in proceedings of CoNEXT, December 2006.
8. R. Alshammari and A. Nur Zincir-Heywood. “A Flow Based Approach For Ssh Traffic Detection”, Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on.
9. Callado, A.; Kamienski, C.; Szabo, G.; Gero, B.; Kelner, J.; Fernandes, S.; Sadok, D.; A Survey on Internet Traffic Identification; Communications Surveys & Tutorials, IEEE Volume 11, Issue 3, 3rd Quarter 2009.
10. http://www.openssh.com/
11. MTU: RFC 879.
12. http://www.caida.org.