This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2010 proceedings

Identification of traffic flows hiding behind TCP port 80

Alberto Dainotti, Francesco Gargiulo
Department of Computer Science and Systems
University of Napoli "Federico II"
{alberto, francesco.grg}@unina.it

Ludmila I. Kuncheva
School of Computer Science
University of Bangor
l.i.kuncheva@bangor.ac.uk

Antonio Pescapè, Carlo Sansone
Department of Computer Science and Systems
University of Napoli "Federico II"
{pescape, carlosan}@unina.it

***Abstract*— Beyond Quality of Service and billing, one of the most important applications of traffic identification is in the field of network security. Despite their simplicity, current approaches based on port numbers are highly unreliable. This paper proposes an identification approach, based on a cascade of decision trees. The approach uses the *sign pattern* and *payload size* of the first four packets in each flow, thus remaining applicable to encrypted traffic too. The effectiveness of the proposed approach is evaluated on five real traffic traces collected in different time periods and over four different networks. The obtained overall accuracy gives us grounds to consider the adoption of this approach as stand-alone in on-line platforms for network traffic identification or in combination with classical firewall architectures.**

I. Introduction

In the last few years, networking research has focused on network traffic classification, that is, associating traffic flows to the applications that generated them [1]. This research stream answers a problem unforeseen when the *Internet* protocols were first designed. Originally, all network applications used known protocols and transport-level ports that easily allowed their identification. This situation changed a few years back [2], [3]. The number of network applications using proprietary undisclosed protocols has grown at an incredible rate (Skype, P2P-IPTV); the typical association application/port is often forged; in some cases traffic is encrypted, whereas sometimes it is encapsulated into traditional protocols. Beyond the need to understand which kind of traffic is carried on Internet links, the identification of traffic hidden in flows using well-known ports represents a challenging task. To answer this challenge, new approaches to traffic identification are needed. By traffic identification here we mean identification of a particular (or a group of) applications of interest.

In this paper we propose a novel application of an identification technique based on packet-level information and exploiting behavioral characteristics of different applications. Specifically, we propose the use of both the *sign pattern* and the *sizes* of the first four packets of each flow to label the flow as accepted (identified) or rejected. The accepted class is the protocol/application conventionally associated with the respective port number. The rejected class is related to applications that try to hide their presence, usually with the purpose to circumvent network usage/security policies. The proposed approach is aimed at a high accuracy of identification, being at the same fast and universal. First, it uses the direction signs and the sizes of only the first four packets of each flow (targeted to work online), and second, it does not need to access the payload of the packets (does not affect privacy and works with encrypted packets).

The rest of the paper is organized as follows. Section II discusses briefly our motivation. Section III provides details about the techniques at the base of our identification approach. Section IV describes the dataset and the measurement approach used in the experimental validation. We show results of identification of "port 80" traffic in Section V. Section VI concludes the paper.

II. Motivation and Related Work

Even if commonly considered unreliable, the classification/identification approach known as *port-based* is still used today for online monitoring. Its advantages are simplicity and speed, as it checks only a single packet-header field. Besides, in some real-world situations there are no effective alternatives. An immediate alternative proposed in the literature (and promptly adopted by the industry) are the *payload-based* approaches based on the inspection of the transport-level packet payload (the data produced by the application). These techniques usually compare packet contents against known signatures of application-level protocols. Such techniques were initially considered very reliable, and were used to build reference data in the evaluation of novel classification approaches [4]–[6]. Today, however, their reliability and applicability are undermined by a number of factors. First, undisclosed proprietary protocols and techniques of protocol obfuscation (e.g., eMule/eDonkey) are continuously arising. Second, several new-generation applications (e.g., instant messaging, file sharing) make use of traditional protocols (e.g. HTTP) to encapsulate their traffic, which deceives the *payload-based* classifiers into erroneously associating the traffic to the encapsulating protocol. Third, new-generation applications (e.g., Skype) use packet-payload encryption techniques. In addition, network-level (e.g., IPSEC) and application-level (e.g., ssh) encryption tunnels are being increasingly used in the Internet. Even when they are feasible, *payload-based* approaches face further difficulties: (i) payload inspection requires accessing all user-transmitted data, which may breach privacy laws in

[0]This work has been supported by the European Communitys 7th Framework Programme (FP7/2007-2013) under Grant Agreement No. 216585 (INTERSECTION Project) and Grant Agreement No. 225553 (INSPIRE Project).

978-1-4244-6404-3/10/$26.00 ©2010 IEEE

some countries; (ii) the computational resources required to inspect the entire content of the packets is usually quite high, making it difficult to deploy such techniques when the traffic volume is large. Because of the growing problems with the *payload-based* approaches, new *statistical-based* classification approaches have been proposed that do not need access to packet content. These approaches use flow characteristics as features to train classifiers from the state-of-the-art machine learning. The explosion of high-quality scientific literature in this field [7]–[18] testifies the great interest in researching novel and accurate techniques for traffic classification. It has been demonstrated that the *statistical-based* approaches can achieve high accuracy, and that they appear to be the most promising approaches against protocol obfuscation, encapsulation, and encryption [14], [19]. In this paper we consider the traffic classification technique presented in [20] and we build a novel approach for the problem of identifying traffic flows that use well-known ports (in this case TCP port 80) while not being generated by the officially associated application protocol (in this case HTTP). To demonstrate the feasibility and applicability on the field of such approach we extend the datasets used in [20] by adding recent traffic traces in order to perform temporal cross-testing.

III. The Identification Approach

The requirement for operational speed brings in the idea of a classifier selection ensemble where only one of a set of experts has to make a decision [20], [23]. The ensemble consists of member classifiers (*experts*) and an *oracle* that authorises one of the classifiers to pass its decision as the ensemble decision. Generally speaking, the oracle may have pre-defined regions of competence for the classifiers [24] or dynamically allocated regions [25]. Gargiulo *et al.* [20] propose to use the port number as the oracle determining the regions of competence. The directions and sizes of the first four packets of the TCP flow are then used as the features in a further 2-stage classifier (Figure 1). The features and the modular architecture were chosen so that the classification is both fast and accurate, and new modules can be trained and added to the system without re-training any already trained part.

A. The Features

According to well established results present in literature [10], [11], [26], [27], we propose to use only the payload sizes and directions of the first few packets. For example, in [11] it is shown that the payload sizes of only the first 4 packets are discriminant features allowing to classify different applications, also including Peer-to-Peer applications. More precisely, we use the following features (see Section IV for details on feature extraction):

- x_0, the port number;
- x_1, x_2, x_3, x_4, the directions of the first four packets, $x_i \in \{0, 1\}$, where 0 means that the packet is transferred from server to client, and 1, from client to server;
- s_1, s_2, s_3, s_4, the payload sizes of the first four packets, where s_i are positive integers. As in [26], we do not consider packets without payload because they are related to connections state information.

TABLE I
SUMMARY OF TRAINING DATA FROM UNIBS.

Signs	Protocol and port number					
	POP3	FTP	SMTP	msn	BitTorr	HTTP
123 4	110	21	25	1863	6881	80
000 0	0	138	16	0	0	3
000 1	1	75	55	0	0	0
001 0	21	216	543	0	0	0
001 1	0	0	4	0	1	0
010 0	749	21	604	1	0	0
010 1	18823	5845	18186	0	1	0
011 0	17	1	18	0	1	0
011 1	0	0	1	0	0	0
100 0	0	0	0	328	23	5348
100 1	0	0	0	30	520	240
101 0	0	0	0	660	3609	826
101 1	0	0	0	4	753	12
110 0	0	0	0	1	8	427
110 1	0	0	0	0	87	76
111 0	0	0	0	0	9	108
111 1	0	0	0	0	45	23

B. Stage 1: Sign Pattern Filter

To illustrate the system we use a data set consisting of network traffic traces at the University of Brescia, Italy (UNIBS) [26]. The known protocols in the training data are: POP3, SMTP, HTTP, msn, FTP and BitTorrent. Table I shows a summary of the training data as distributed across the 16 possible patterns of signs $[x_1, x_2, x_3, x_4]$, from 0000 to 1111.

The table shows that groups of protocols can be distinguished by the sign patterns. For example, protocols msn (1863), BitTorrent (6881) and HTTP (80) hardly ever begin with a packet from server to client ($x_1 = 0$). The table suggests that the sign patterns can be used to filter out very quickly flows that do not match the pattern of the class they are supposed to be a part of. In Figure 1, this is labeled as the *sign pattern filter*. In this paper we focus on the TCP traffic on port 80, so flows with patterns beginning with $x_1 = 0$ will be rejected by the filter. Next, using the training data, we can choose a rejection threshold, of say, 2%, and filter out all sign patterns where the number of flows is below the threshold. With this filter in place, the "allowed" combinations of signs for the HTTP protocol (80) are 1000, 1001, 1010, and 1100. All other protocols will be rejected by the sign pattern filter.

C. Stage 2: Decision Tree classifier using payload sizes

A separate classifier is then trained for each sign combination that passes through the sign filter. Here, each classifier has to solve a two-class problem: match versus mismatch of the protocol/application conventionally associated with the respective port number. We chose the *Decision Tree* [29] classifier, since its classification speed makes it very effective for an online implementation [28] and it does not assume any type of probability distribution of the data [20]. In addition, since the training of the decision trees is made offline, learning times are not significant. Moreover, for the binary decision trees we used, the training time complexity is $O(4N^2 * \log N)$ [29], where N is the size of the training set.

The decision process of a Decision Tree classifier is intuitive, since it can be traced as a sequence of simple decisions.

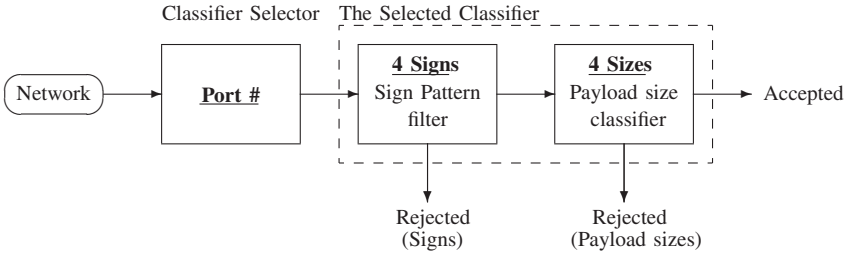

Fig. 1. The generic classifier ensemble architecture.

The first decision is made at the root; depending on the answer, a branch is selected and the child node is visited. Another decision is made at this node, and so on, until a leaf is reached. The leaf contains a single class label, which is assigned to the object being classified. In our case the C4.5 algorithm was employed for constructing the Decision Tree classifiers. We used the Weka implementation, called J48[1].

The choice of a Decision Tree classifier can be justified by the following example. Figure 2 shows the scatterplot of a dataset of traffic traces taken from the University of Brescia, Italy (see Section IV for the dataset description). The data is filtered so that only flows with sign pattern $[x_1, x_2, x_3, x_4] = 1010$ are displayed. The (x, y) coordinate axes are the first two size features, s_1 and s_2, respectively. The figure shows three protocol classes: BitTorrent (3609 flows), HTTP (826) and msn (660). Class HTTP seems the predominant class in Figure 2, however, this is not the case. Classes BitTorrent and msn are extremely dense, and are located towards the bottom left corner of the scatterplot. The distribution of the classes suggests that the classification regions will be best delineated using borders parallel to the coordinate axes. This is exactly the type of regions constructed by the Decision Tree classifier.

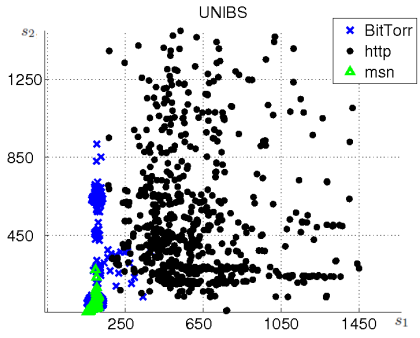

Fig. 2. Scatterplot of the UNIBS training data in the plane of the first two size features, s_1 and s_2.

IV. Dataset and Feature Extraction

To validate the proposed approach we used training datasets from three different institutions: *University of Brescia in Italy (UNIBS)* [32], *Lawrence Berkeley National Laboratory (LBNL)* [33] and *Cooperative Association for Internet Data Analysis (CAIDA)* [31]. A summary of the content of the three data sets used to train our system is given in Table II. As testing set we used traces from *University of Napoli in Italy (UNINA)* [34]. From this network we collected and used traffic traces related to two different time periods, 2004 (hereinafter denoted as *UNINA2004*) and 2009 (hereinafter denoted as *UNINA2009*). Details about the flows composing these traces are reported in Table III.

TABLE II
Number of flows in the three training data sets.

Protocol	Port	UNIBS	CAIDA	LBNL
POP3	110	19611	9591	1172
SMTP	25	19427	11831	20825
HTTP	80	7063	5930	81984
FTP	21	6296	1652	–
BitTorrent	6881	5057	–	–
msn	1863	1024	–	–
netbios-ssn	139	–	4575	–
HTTPS	443	–	25427	18013
oms	4662	–	–	1716
IMAP4	993	–	–	7677

TABLE III
Number of flows in the UNINA data sets used for testing.

Protocol	Port	UNINA2004	UNINA2009
HTTP	80	506795	144042
non-HTTP	80	2245	803

In this work we focus our experiments on the identification of HTTP traffic flowing through port TCP 80. Thus, during the training phase, for each sign pattern we train the corresponding payload size classifier (a Decision Tree) assigning all the HTTP flows to one class (the one corresponding to traffic to be accepted), and all the other flows (e.g. from msn, BitTorrent, etc.) that match the considered sign pattern as the other class (traffic to be rejected). For example, in the case of the 1010 combination for the UNIBS dataset we train the classifier with HTTP against msn and BitTorrent (see Table I).

To extract the nine features we used ***TIE (Traffic Identification Engine)*** [21] [22], an open-source multi-classifier system whose architecture is shown in Figure 3.

For this work we used TIE for (i) processing and filtering traffic traces, (ii) aggregating packets into sessions, and (iii)

[1]Weka is an open source collection of data-mining tools and is freely available at the website `http://www.cs.waikato.ac.nz/ml/weka`.

extracting features. The features produced by TIE have been fed to a prototype implementation of the identification approach described in Section III. Moreover, we used TIE with a classification plugin based on a payload-inspection technique in order to establish the "ground-truth" of the given traces. This allowed us to label each flow and to evaluate the accuracy of our approach (see Section V). We looked at the payload content using the *TIE-L7* plugin module, which implements the Linux L7-filter classification technique [30].

In the experiments presented we focused on traffic on TCP port 80. The *Packet Capture* TIE module filters out all traffic not pertaining to the port, whereas the TIE module named *Session Builder* is responsible for aggregating the remaining packets into bidirectional flows (*biflows*). That is, we consider the common definition of flow tuple while taking into account traffic in both directions: upstream and downstream. The upstream direction is the one of the first packet observed. Moreover, because we are examining TCP traffic, instead of a timeout value we use simple heuristics based on SYN, FIN, RST flags in TCP headers, in order to approximate TCP connections (as described in [21]).

The *Feature Extraction* module is responsible for extracting classification features from each biflow. In order to take into account only properties related to the application, we record the sizes of the transport-level payload, excluding pure TCP packets that do not carry application-level data (e.g., empty ACK packets). The payload sizes are stored in the order they are observed. A sign is added depending on the packet direction, plus for upstream and minus for downstream. Each biflow is assigned a session_id, which can later be used to manually examine the biflow, or to process again the same traffic trace using TIE classifiers and checking the results (as in Section V).

V. Experimental Results

In this section we show the results obtained with the proposed identification architecture. In particular, we want to demonstrate that if we train the system with traffic traces taken from different sites (spatial invariance) and in a different time (temporal invariance) we can correctly accept HTTP traffic and reject non-HTTP traffic[2].

To show the results we choose the following metrics:

- **Overall Accuracy**: The percentage of correctly classified flows.
- **HTTP Accuracy**: The percentage of the correctly classified HTTP flows out of all true HTTP flows (sensitivity).
- **non-HTTP Accuracy**: The percentage of the correctly classified non-HTTP flows out of all non-HTTP flows (specificity).

For assessing the effectiveness of the proposed approach in different time periods (temporal invariance), we carried out cross-testing using UNINA2004 and UNINA2009 traces. In order to verify the spatial invariance of the approach, we train the system with LBNL, CAIDA and UNIBS traces, and then test the system with both UNINA2004 and UNINA2009 traces.

Table IV is obtained by training and testing the system by using a 10-fold cross validation protocol. That is, the whole dataset is divided into 10 folds; 9 of them are used to train the classifiers and the last fold is used for testing. This is carried out for all 10 folds and results are reported as average accuracies. The table shows high accuracy for UNINA2004, and hints about the diversity of non-HTTP traces that might have caused the low specificity for UNINA2009.

TABLE IV
Results obtained by training and testing the system with UNINA2004 and UNINA2009.

	Overall Accuracy	HTTP Accuracy	non-HTTP Accuracy
UNINA2004	99.97%	99.99%	96.08%
UNINA2009	99.97%	99.99%	86.23%

Tables V and VI report the results obtained by training the system with LBNL, CAIDA and UNIBS traces, and testing it with the other two traces. Both tables indicate that the recognition rate of the non-HTTP protocol depends on the training set. The worst results in rejecting non-HTTP flows are obtained when LBNL traces are used for training and the system is tested on UNINA2009 (85.45%). This is due to the fact that class non-HTTP is not very well represented in the LBNL data. Figure 4 shows a scatterplot of the UNINA2009 data for the four "allowed" sign patterns for port 80. The non-HTTP protocols are marked with green triangles. The misclassified protocols are circled. The figure demonstrates a degree of mismatch between the training (LBNL) and the testing (UNINA2009) data. It should be noted however, that the representation in the figure may be misleading because it does not reflect the density of the data. The plot for sign 1000, (a), for example, contain 47616 traces, of which 11 non-HTTP. There is only one mistake in the non-HTTP class (accepting a non-HTTP protocol), which amounts to 91% specificity. The highlighted mistakes are only a fraction of the true HTTP class that were wrongly rejected (185 biflows in subplot (a), equivalent to 0.39%).

As with Table IV, there is a decline in the correct recognition rate of non-HTTP traffic from 2004 to 2009. Again, this may be explained with the hypothesis that some of the new non-HTTP traffic biflows are more similar to normal HTTP compared to the ones in the 2004 data. This notwithstanding, the obtained results remain very good since in the worst case over 85% of non-HTTP flows are rejected. In order to

[2]It is worth noticing that while HTTPS has been considered for training the system (see Table II), we did not consider HTTPS traffic in the traffic to be recognized as HTTP because it uses a different port. The approach we propose in the paper was tested on port TCP:80.

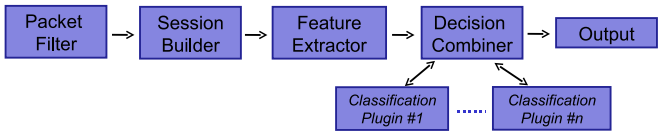

Fig. 3. Overall Architecture of TIE.

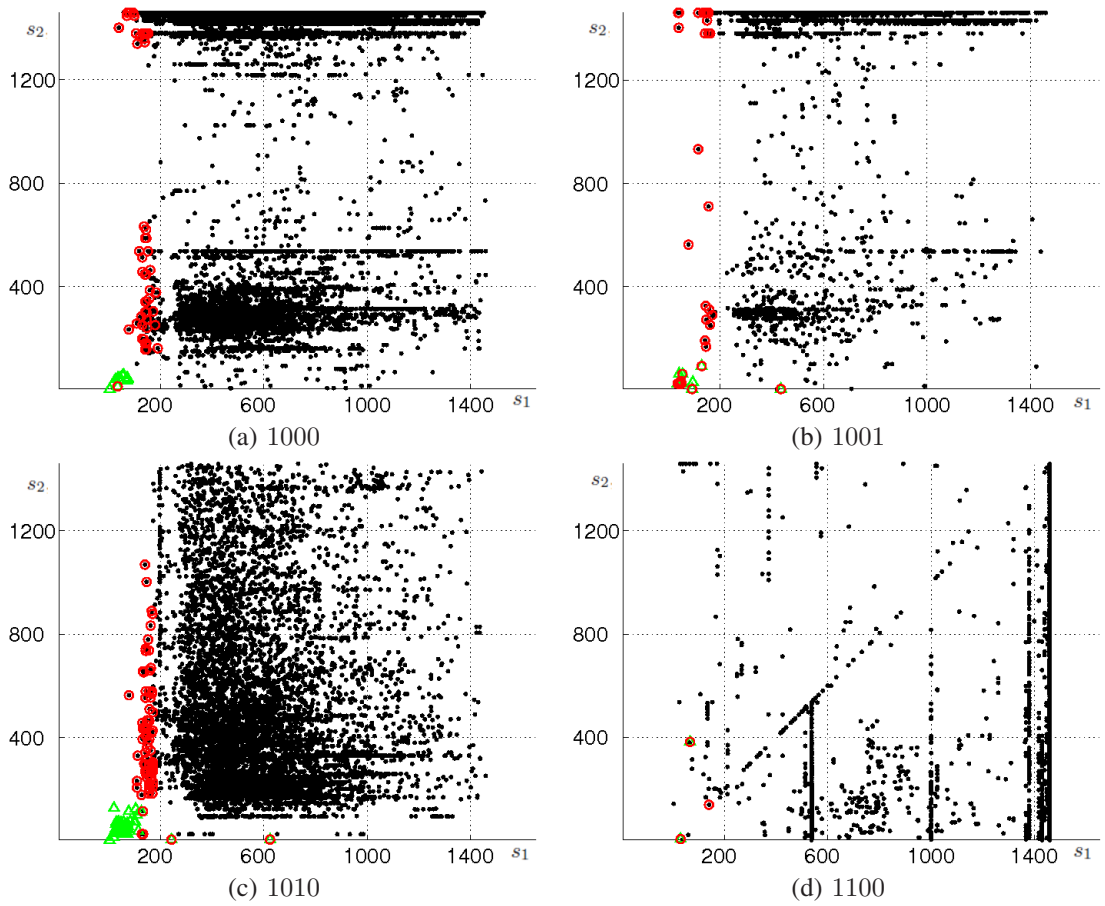

Fig. 4. Scatterplot of the UNINA2009 using LBNL training data in the plane spanned by the first two size features, s_1 and s_2. HTTP biflows are shown in black while non-HTTP are shown in green. Misclassified samples are highlighted.

improve the performance the Decision Tree classifiers may be re-trained with new counterexamples. An advantage of the chosen architecture is that it allows us to do that for any of the classifiers without changing the rest of them.

TABLE V

RESULTS OBTAINED BY TESTING THE SYSTEM WITH UNINA2004.

Training Dataset	Overall Accuracy	HTTP Accuracy	non-HTTP Accuracy
LBNL	99.69%	99.73%	88.96%
CAIDA	99.25%	99.25%	97.84%
BRESCIA	96.45%	96.44%	99.64%

TABLE VI

RESULTS OBTAINED BY TESTING THE SYSTEM WITH UNINA2009.

Training Dataset	Overall Accuracy	HTTP Accuracy	non-HTTP Accuracy
LBNL	99.26%	99.28%	85.45%
CAIDA	98.82%	98.83%	92.73%
BRESCIA	99.21%	99.22%	94.55%

In order to better assess the approach here presented and to further investigate the results obtained, we analyzed traffic flowing through port TCP 80 that was labeled by our identification system as 'rejected'. Firstly, we added a feature in TIE to examine the first few bytes of payload carrying TCP payload exchanged in each biflow. This allowed us to perform a preliminary automated examination of all the biflows and to verify the results of the identification by easily checking application-level packet content. In addition, we manually inspected the biflows, mainly focusing on what was recognized as non-HTTP by the classifier. First of all, such analysis confirmed that all the correctly accepted biflows were actually related to HTTP traffic. For example we observed that almost 94% of the biflows in UNINA2004 started with a standard HTTP GET request, 4% with a POST request, etc. On the other side, we discovered that several 'rejected' biflows we generated by peer-to-peer application-level protocols as eDonkey, Bittorrent, and WinMX. Some of them started with a byte not corresponding to an alphabetic character. Inside this category, most of them started with the byte 0xe3. As reported by Karagiannis *et al.* in [6], this is the first byte exchanged by peers opening a communication session based on the eDonkey2000 protocol (used by the eDonkey and eMule file-sharing applications). Moreover, in both UNINA2004 and UNINA2009 traces, up to 50% of the non-HTTP biflows could not be ascribed to a specific application using either automated or manual payload inspection. However, we manually verified that these biflows did not exchange any HTTP traffic; we therefore conclude that such traffic is generated by applications

using undisclosed proprietary protocols.

The whole analysis described here confirms that the identification approach proposed in this work is very effective in correctly discriminating real HTTP traffic using the well-known port TCP 80. Finally, we observe that in the traces the non-HTTP traffic represents a not negligible portion of the captured traffic. Indeed, after filtering our traces by removing biflows related to non-HTTP traffic, about 5% of the packets were discarded (both in the UNINA2004 and UNINA2009 trace). Moreover, it must be observed that on the UNINA network there were no rules enforced to prevent traffic on non-standard ports. Therefore most of the connections masquerading as HTTP were probably due to the configuration of external peers located in networks where port-based traffic filtering was strictly enforced. It is reasonable to hypothesize that if this was also the case of the UNINA network, then such masquerading traffic would have covered an even higher percentage.

VI. Conclusion

In this work we examined the ability of a classifier ensemble to identify traffic flows that do not belong to their declared class. The system takes the direction signs of the first four packets carrying payload and filters out the most improbable flows. The remaining flows have "acceptable" sign patterns. A decision tree classifier is trained for each sign pattern. Here we focused on TCP on port 80, trying to separate true HTTP traffic from non-HTTP traffic flowing through the same port (e.g. in order to circumvent network policies). We imposed four acceptable sign patterns: 1000, 1001, 1010 and 1100. A Decision Tree classifier is designed for each of them. We found that the system is very accurate when trained and tested on data coming from the same distribution (tested through cross-validation on traces from the University of Napoli - UNINA2004 and UNINA2009). Furthermore, the system exhibits very high accuracy in cross-testing, i.e., trained on one network and tested on another. We verified this by training the system on three different data sets (LBNL, CAIDA and BRESCIA) and testing it with UNINA2004 and UNINA2009. We looked in more detail in the worst case (accuracy 85.45%) where the system was trained on LBNL and tested on UNINA2009. It seems that the LBNL data did not have a non-HTTP class enough representative to train the system properly. The high overall accuracies in the cross-testing demonstrate what we call the "spatial invariance" of the system. In addition, the system shows "time invariance" in that the accuracy did not drop dramatically from 2004 to 2009 even though some decline was observed. Finally, due to its design and to the use of Decision Trees, the system is very fast, and it can be used online. Another advantage of a classifier based on Decision Trees is that it can be seen as a set of simple decision rules that can be easily interpreted by a domain expert. A further research direction will be a deeper analysis of such decision rules to better understand the behavior of the proposed traffic identification system. We will also apply the proposed approach to other well-known ports.

References

[1] M. Mellia, A. Pescapé, L. Salgarelli, "Traffic classification and its applications to modern networks", Computer Networks, Vol. 53, Issue 6, 759-760, April 2009.

[2] T. Karagiannis, A. Broido, N. Brownlee, KC Claffy, M. Faloutsos. "Is p2p dying or just hiding?". In IEEE Globecom, 2004.

[3] A. Moore and K. Papagiannaki. "Toward the accurate identification of network applications". In PAM, April 2005.

[4] L7-filter, Application Layer Packet Classifier for Linux. http://l7-filter.sourceforge.net.

[5] V. Paxson, "Bro: A system for detecting network intruders in realtime". In Computer Networks, pages 23-24, 1999.

[6] T. Karagiannis, A. Broido, M. Faloutsos, KC Claffy. "Transport layer identification of p2p traffic". In ACM IMC, October 2004.

[7] A. Moore, D. Zuev. "Internet traffic classification using bayesian analysis techniques". In ACM SIGMETRICS, June 2005.

[8] J. Erman, A. Mahanti, M. Arlitt. "Internet traffic identification using machine learning". In IEEE Globecom, November 2006.

[9] S. Zander, T. Nguyen, G. Armitage "Automated traffic classification and application identification using machine learning" IEEE LCN, Nov. 2005.

[10] T. Auld, A. Moore, S. Gull. "Bayesian neural networks for internet traffic classification". IEEE Trans. Neural Networks, 18(1):223-239, Jan. 2007

[11] L. Bernaille, R. Teixeira, K. Salamatian. "Early application Identification". In ACM CoNEXT, December 2006.

[12] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, C. Williamson. "Offline/realtime traffic classification using semi-supervised Learning". In IFIP Performance, October 2007.

[13] T. Karagiannis, K. Papagiannaki, M. Faloutsos. "Blinc: Multilevel traffic classification in the dark". In ACM SIGCOMM, August 2005.

[14] L. Bernaille and R. Teixeira. "Early recognition of encrypted Applications". In PAM 2007, pages 165-175.

[15] S. Zander, T. Nguyen, G. Armitage. "Self-learning ip traffic classification based on statistical flow characteristics". In PAM, 2005.

[16] N. Williams, S. Zander, G. Armitage. "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification". ACM SIGCOMM CCR, 36(5):7-15, October 2006.

[17] L. Salgarelli, F. Gringoli, T. Karagiannis. "Comparing traffic classifiers". ACM SIGCOMM CCR, 37(3):65-68, 2007.

[18] A. Dainotti, W. De Donato, A. Pescapé, P. Salvo Rossi, "Classification of Network Traffic via Packet-Level Hidden Markov Models", IEEE GLOBECOM 2008 - Dec 2008, New Orleans (LA, USA)

[19] C. V. Wright, F. Monrose, G. M. Masson. "On inferring application protocol behaviors in encrypted network traffic". Journal of Machine Learning Research, 7:2745-2769, December 2006.

[20] F. Gargiulo, L.I.Kuncheva, C. Sansone, "Network Protocol Verification by a Classifier Selection Ensemble". In Proc. of the 8th Conference on Multiple Classifier Systems (MCS), 2009.

[21] A. Dainotti, W. De Donato, A. Pescapé, "TIE: a Community-Oriented Traffic Classification Platform", TMA'09, May 2009, Aachen (Germany)

[22] http://tie.comics.unina.it

[23] L.I. Kuncheva. Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, 2004.

[24] L.A. Rastrigin, R.H. Erenstein. Method of Collective Recognition. Energoizdat, Moscow, 1981. (In Russian).

[25] K. Woods, W.P. Kegelmeyer, K.W. Bowyer. Combination of multiple classifiers using local accuracy estimates. IEEE Trans. Pattern Anal. Mach. Intell., 19(4):405410, 1997.

[26] A. Este, F. Gargiulo, F. Gringoli, L. Salgarelli, C. Sansone. "Pattern recognition approaches for classifying ip flows". In volume 5342 of Lecture Notes in Computer Science, pages 885-895. Springer, 2008.

[27] E.P. Freire, A. Ziviani, R.M. Salles. "On metrics to distinguish skype flows from http traffic", LANOMS 2007, pages 57-66, 2007.

[28] N. Williams, S. Zander, G. Armitage: "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification". ACM SIGCOMM CCR, 36(5), 7-15 2006.

[29] R.O. Duda, P.E. Hart, D.G. Stork, "Pattern Classification", 2nd edn. Wiley, Chicheste, 2000.

[30] http://l7-filter.sourceforge.net/

[31] http://www.caida.org/data/

[32] M. Crotti, M. Dusi, F. Gringoli, L. Salgarelli. "Traffic classification through simple statistical fingerprinting", ACM SIGCOMM CCR, 2007

[33] http://ee.lbl.gov/anonymized-traces.html

[34] http://www.grid.unina.it/Traffic/Traces