Detecting Skype flows in Web traffic

Emanuel P. Freire,* Artur Ziviani,† and Ronaldo M. Salles*
*IME – Military Institute of Engineering
Praça General Tibúrcio, 80 – 22290-270 – Rio de Janeiro, RJ, Brazil
Email: salles@ieee.org
†LNCC – National Laboratory for Scientific Computing
Av. Getúlio Vargas, 333 – 25651-075 – Petrópolis, RJ, Brazil
Email: ziviani@lncc.br

***Abstract*—Network managers face nowadays a challenging problem to detect traffic from Skype, a very popular application for VoIP communications. If no restrictive firewalls are adopted, Skype uses UDP as its preferred transport protocol, but it is known that due to its high capacity of adaptation, Skype can operate behind many firewalls and network proxies without user configuration. Behind restrictive firewalls, Skype uses Web TCP ports (80 or 443) as a fallback mechanism to delude firewalls and other network elements. This strategy renders Skype traffic disguised as Web traffic quite difficult to detect by network operators. In this paper, we propose a method to efficiently detect Skype flows hidden among Web traffic. We validate our proposal using real-world experimental data gathered at a commercial Internet Service Provider (ISP) and an academic institution. Our experimental results show a performance of around 90% detection rate of disguised Skype flows with a false positive rate of only 2%, whereas a 100% detection rate of Skype flows in Web traffic is achieved with a false positive rate limited to only 5%. We also evaluate the feasibility of our proposal in a real-time Skype detection scenario.**

I. Introduction

An efficient classification of the application protocol responsible for a given traffic is a fundamental issue in network management. Typically, network operators rely on TCP/UDP port numbers to allow or deny access into their domains, following a list with registered and well known TCP/UDP port numbers provided by the Internet Assigned Numbers Authority (IANA) [1]. Recently, however, the efficient classification of the application protocol responsible for a given traffic has became a more challenging problem, mainly because TCP/UDP port numbers are no longer a reliable information source to identify the application responsible for a given network traffic [2]. There is usually no control to ensure that an application only uses its reserved ports to send or receive data, and with the increasing use of network elements such as firewalls, NAT boxes and proxies, network applications evolved to operate in different environments with minimal user configuration, for example, dynamically choosing a TCP/UDP port number.

In order to optimize the use of network resources, very restrictive firewalls are commonly adopted by network managers in many organizations using port numbers to select some applications to receive priority treatment or to be blocked. Although being very restrictive, such firewalls are unlikely to block Web traffic because it is usually perceived as a basic service considered essential for the most part of organizations. As HTTP is typically the most popular application protocol in the Internet, it has become usual to find applications—such as recent P2P file sharing systems [3] as well as media streaming [4] or VoIP calls—using TCP ports 80 (HTTP) or 443 (HTTPS) for non-HTTP traffic to delude firewalls or other network elements.

Among the applications that adopt the strategy of disguising their flows as Web traffic to delude firewalls and other network elements, Skype is of particular importance due to its huge popularity. Skype is a very popular voice over IP (VoIP) application with a proprietary closed-source protocol. It can easily work in different network environments, as it can automatically detect network characteristics and use other computers to relay its traffic. It is also known that Skype can delude a network firewall by using Web ports to establish communication with other Skype peers. This strategy is adopted by Skype as a fallback mechanism in the case of other strategies fail to get through a restrictive firewall. Such a strategy renders Skype traffic disguised as Web traffic quite difficult to detect by network operators.

In this paper, we are particularly interested in investigating the detection of Skype flows in Web traffic. In a previous work [5], we have investigated metrics to distinguish Skype flows from Web traffic using two Goodness-of-Fit tests, the Kolmogorov-Smirnov and the Chi-square test. We build upon the metrics proposed in that work to develop a novel detection methodology to automatically classify supposed Web traffic into either legitimate Web browsing or Skype VoIP flows using Web ports. One can also classify these flows searching for Skype patterns in captured data, but such approach is likely to be more dependent of Skype version or specific traffic details. We consider our approach more robust because it can detect Skype traffic without a search for particular Skype patterns or signatures and without regarding payload information. We intend to use a method in our detection that could capture general characteristics of VoIP flows such as the regular flow of small packets.

The recent work by Bonfiglio et al. [6] shows two different approaches to identify Skype traffic in TCP or UDP flows. Their first method uses some Skype-specific information for UDP detection and it uses the Pearson's Chi-Square test to verify if payload data appears to be random for UDP or

TCP detection. We adopt the Chi-Square test, a variation of Pearson's Chi-Square test, and in a different context: to compare flows parameters with empirical data derived from real Web flows. Our methodology was specifically designed to deal with the HTTP protocol and we believe it can be extended to detect other applications that may be using HTTP ports as well.

We evaluate our Skype detection methodology using experiments with real-world data gathered at a commercial Internet Service Provider (ISP) and an academic institution. Our results show that the Chi-square test is more likely to achieve better results than the Kolmogorov-Smirnov test for Skype detection. The experimental results also show that the proposed methodology achieves good performance in detecting Skype flows in Web traffic. Such a good performance may be illustrated in our experimental results by the observation of a 90% detection rate of disguised Skype flows with a false positive rate of only 2%, whereas a 100% detection rate of Skype flows in Web traffic with a false positive rate limited to only 5%. We also evaluate the feasibility of our proposal in a Skype detection scenario with real-time monitoring. In such case, network managers could choose to perform an immediate action after detection, for example, blocking all traffic identified as Skype from his network or giving such traffic a priority link.

The remainder of this paper is organized as follows. Section II briefly discusses related work in the classification of applications and Skype analysis. In Section III, we present our methodology to distinguish Skype flows from Web traffic. Based on this methodology, we propose a Skype detection system in Section IV. Section V presents our experimental results evaluating the proposed detection system. Finally, we conclude and discuss future work in Section VI.

II. Related Work

One of the first works to characterize network traffic anomalies was the study of Barford and Plonka [7]. The authors defined three basic types of network anomalies: anomalies caused by network operations such as device malfunction or configuration changes, anomalies caused by abuses like denial of service attacks and anomalies caused by users such as flash crowd events. In the work of Lakhina et al. [8] a more detailed characterization is presented and the use of the subspace method is proposed for anomaly diagnosis. Other work from the same authors [9] showed the use of entropy as a summarization tool. In [10], the authors searched for anomaly detection in large scale networks using traffic matrices and a Kalman filter to characterize a model describing the normal behavior and compare it with the observed behavior. Some different methods were tested and the results were compared using ROC curves. In this work, we used the approach of building a model of the "normal" behavior and compare it with the observed behavior and we also used ROC curves for evaluation, but we are interested in detecting a specific kind of traffic anomaly: the use of the HTTP port by applications to send non-HTTP traffic. We used a Web workload model developed in a previous work [5], which was based on earlier works in this subject [11], [12].

Application identification and classification has been the focus of recent related work [2], [13]–[16]. Karagiannis et al. [2] develop a traffic classifier which operates in various levels of detail and without looking to the payload information. In [13], authors search for an accurate application identification methodology composed of several steps. It involves manual intervention and a previous knowledge of the protocols' behavior. Bernaille et al. [14] investigate a fast but not so accurate traffic classification method based on the size of the n first packets found within a given flow. Ma et al. [15] propose a framework for unsupervised protocol inference, comparing three classification techniques for protocol classification. The work of Won et al. [16] proposes a hybrid approach for application traffic identification based on signature matching on the initial packets or payload bytes and session information. Nevertheless, to the best of our knowledge, no method was specifically designed to deal with protocol anomalies in Web traffic.

Due to the huge popularity Skype achieved in the last few years, analysis of the Skype protocol and characteristics have been the focus of recent related work. Baset and Schulzrinne [17] present an analysis of the Skype behavior during login, call establishment, firewall/NAT traversal, and other operations. Guha et al. [18] performed in 2005 five experiments to analyze Skype traffic characteristics and better understand its operation. Suh et al. [19] monitored Skype traffic using relay nodes. They used heuristics and statistical analysis to detect Skype relayed traffic. Ehlert et al. [20] studied Skype network traffic searching patterns and traffic signatures that can allow Skype to be detected. Bonfiglio et al. [6] recently adopted two techniques to detect Skype traffic: one method uses the Chi-Square test while the other is based on Naive Bayesian Classifiers. The results were evaluated with a payload-based classification and the best results were obtained using the two detection methods combined.

In contrast to related work, our proposed Skype detection system is able to distinguish Skype flows in Web traffic without a search for particular Skype patterns or signatures and without regarding payload information. We believe this is a significant contribution as our relatively simple methodology might be extended in future work to develop a more generalized system to detect network anomalies in Web traffic caused by other applications such as video streaming and P2P file sharing.

III. Methodology

The detection process can be subdivided in two steps. First, we define a HTTP Workload Model and capture real Web data to build empirical distributions of some relevant parameters. Then, we capture Web with Skype data, calculate the same relevant parameters for each flow and use a Goodness-of-fit test to decide if the computed parameters are compatible with the empirical distributions derived in the previous step, classifying each flow as Web or Skype. In this section, we

review the Web model and the statistical tests used and present an overview of the Skype program.

A. HTTP Workload Model

We are interested in finding Skype flows hidden among Web flows. Since we avoid relying on program signatures or patterns that can be easily changed, we must define a model for evaluate Web "normal" behavior. In this paper, we build upon the model defined in [5]. This model has the following parameters:

- Web request size;
- Web Response size;
- Interarrival time between requests;
- Number of requests per page;
- Page retrieval time;

B. Goodness of fit tests

In the case where we do not know the underlying distribution of some population, we can use a goodness-of-fit measure to test if a particular distribution can be satisfactory as population model. We used the chi-square test and the Kolmogorov-Smirnov test to distinguish Skype flows from Web traffic. These tests had already been used for anomaly-based intrusion detection [21], [22], or to verify the presence of random payloads in a Skype detection [6]. But in our work, we do not use the chi-square χ^2 value or the Kolmogorov-Smirnov D value to accept or reject the initial hypothesis with a given significance level based in some known distribution. We directly compared the calculated χ^2 and D values with given thresholds to decide if some flow is likely to be Skype or not. This solution can provide more simplicity and flexibility to our program, since we only need to change the threshold values to get a loose classification or a more conservative one.

1) Chi-square test: The χ^2 goodness of fit test, was first investigated by Karl Pearson in 1900 [23]. Basically, it tests a null hypothesis that the observed frequencies of some independent events follow a specified distribution. Suppose we have n observations from a population classified into k mutually exclusive classes and there is some theory or hypothesis which says that an observation falls into class i with probability p_i $(i = 1, \ldots, k)$, so, the number of events expected in class i is $E_i = np_i$. If O_i is the number of events observed in class i, the chi-square statistic χ^2 is the sum over all bins as given by

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}. \quad (1)$$

A large value of the sum indicates that is rather unlikely that the O_i values are drawn from the population represented by the E_i.

2) Kolmogorov-Smirnov test: The Kolmogorov-Smirnov test [24] also tests if a sample comes from a population with a hypothesized distribution. It is based on the maximum difference between two cumulative distributions, $F_0(x)$ and $S_N(x)$. $F_0(x)$ is some specific cumulative frequency distribution function, in our case, the empirical distribution function derived from the training part. $S_N(x)$ is the cumulative step function of a sample of N observations or, in other words, $S_N(x) = c/N$ where c is the number of observations with a value less than x. The Kolmogorov-Smirnov D value is given by

$$D = \max\left(|S_N(x) - F_0(x)|\right). \quad (2)$$

C. Skype characteristics

Skype adopts a proprietary protocol to perform peer-to-peer communication among users. It does not use SIP or other known signaling protocol for VoIP calls and all its traffic is end-to-end encrypted. Skype has the ability to automatically detect network characteristics and choose the best option available to communicate with other Skype peers. As shown in Skype related articles [17], [20], [25], it only uses Web ports as a fallback mechanism, when UDP is not available. In adopting this strategy, Skype can successfully work behind many restrictive firewalls or proxies without user configuration. Skype is also known to have traffic when the program is running but not being used, as it can relay traffic from other hosts. Any computer with sufficient resources might automatically start relaying traffic from other Skype users, but this apparently does not happen in a firewall-restricted computer. Skype also generates traffic to verify if its peers are still active and in other operations such as logins. This traffic does not represent VoIP calls and ideally should not be identified as Skype in our tests.

IV. Detection Process

The first step of the detection process is using a training dataset to characterize a "normal" Web traffic behavior. We capture HTTP full packet traces using the `tcpdump` [26] program, generating dump files. We have developed a software based on the `tcpflow` [27] program to read these dump files and calculate the parameters present in the Web workload model defined in Section III-A. `tcpflow` is a GPL software that can read `tcpdump` captured data and separate each flow present in it. Our software works only with Web traces, it can separate each flow present in a capture file, define Web pages boundaries for each flow, and calculate our parameters for each Web page. A Web page is considered as the complete set of one or more objects in a Web document, normally a HTML file and some images. In this part, while searching for Web pages boundaries, we read HTTP headers to clearly identify a Web request or a Web response and we also compute the inactivity time between Web messages. We must also assure that our data is pure HTTP, so we made a full packet capture to filter all non-HTTP data. After the calculations for all Web flows present in the dump file, the results are combined and we build empirical distributions showed in Section IV-A that will be used in the statistical tests.

The second step is the detection part. Again, we captured Web packet traces using `tcpdump`, but this time only TCP/IP headers were captured. We developed another software to read dump files and calculate each model parameter. This software

TABLE I
WEB TRAINING TRACES CAPTURED.

Trace	Period	Average daily volume
ISP-1	18-20 Jun 2007	45 GB
ISP-2	14-16 Aug 2007	50 GB
ACD-1	14-17 Aug 2007	35 GB

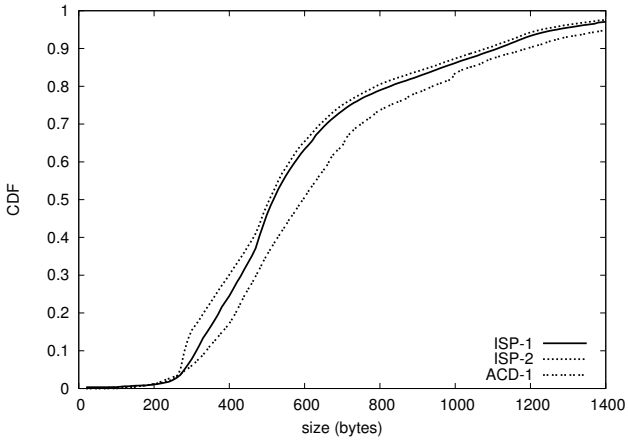

Fig. 1. CDF of the sizes of Web requests.

is different from the program used in the first part because this time the calculations and the division of flows in Web pages are done without examining TCP payload (HTTP headers) information. The procedure for defining Web messages sizes is to consider every MTU-sized packet as a part of the same Web message, if there is not too much inactive time between them. The procedure for defining Web pages boundaries is also based on inactivity time with a fixed threshold.

For each Web page, the five parameters are calculated again, but the results have different treatment. The number of requests per page and the page retrieval time have a single value in each Web page and are somewhat correlated, so we used the number of requests per page as a filter to remove smaller flows. In fact, a Skype flow persists at least for some seconds, so it must have many requests. The other three parameters are represented by a list of values and they are used in Equations (1) and (2) to generate a χ^2 or a Kolmogorov-Smirnov D score. Each parameter generates a score, so in order to make a classification, we have three values that can be compared with thresholds to define if this set of related request-response messages is likely to be Skype or not. For classifying the entire flow, the results obtained for each flow component are combined. If the majority of flow components are identified as Skype, the flow itself is classified as Skype traffic.

A. Training Dataset

We used two types of real-world packet traces, one gathered at a commercial Internet service provider (ISP) and the other originated from an academic institution (ACD). Information about our training traces is shown in Table I. Trace ISP-1 and ISP-2 were captured from two distinct links of the same ISP, located in Niterói, Brazil and with a 2 months interval between

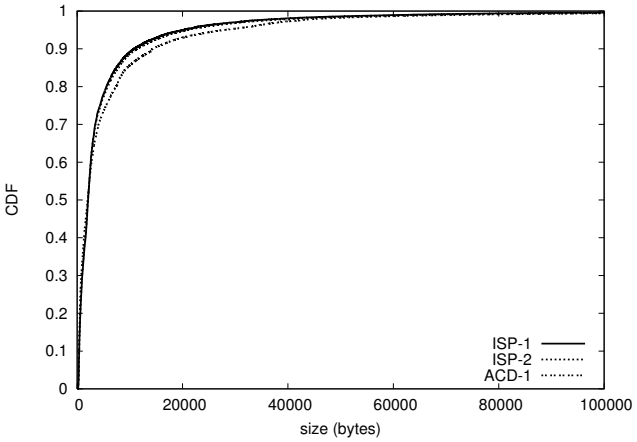

Fig. 2. CDF of the sizes of Web responses.

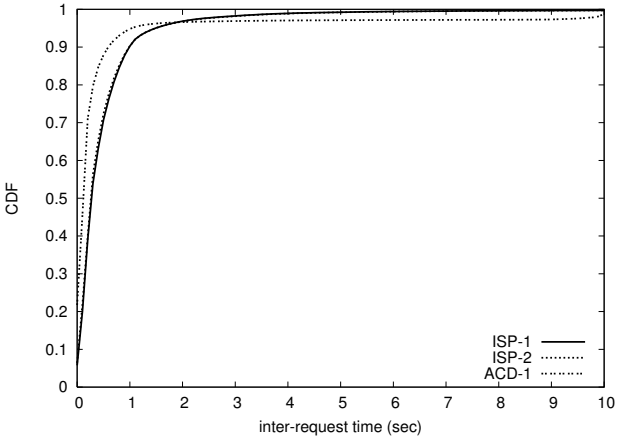

Fig. 3. CDF of the interarrival times of Web requests.

them. Trace ACD-1 was captured from the main Internet link of an academic institution located in Petrópolis, Brazil. The average daily volume represents the amount of Web traffic captured divided by the number of days, not the total average traffic through the link.

Figure 1 presents the Cumulative distribution function (CDF) of the Web request messages sizes for all traces. We observe that Traces ISP-1 and ISP-2 have a very close graph and Trace ACD-1 is slightly inferior. For each trace, the volume of requests larger than 1,500 bytes was less than 0.8% of all computed requests. In Figure 2, we have the CDF of the Web response messages sizes for all traces. Again, Traces ISP-1 and ISP-2 are very similar and Trace ACD-1 is a little inferior before the 40,000 bytes mark. In all traces, the number of Web responses larger than 100,000 bytes was not significant (less than 1%), but their share in volume was around 40% for Traces ISP-1 and ISP-2 and 31% for Trace ACD-1.

Figure 3 shows the CDF for the interarrival times between Web requests. It is the time interval between two consecutive requests for the same Web page, therefore this is a metric for Web pages with two or more requests. As the number of requests in a page increases, more values are generated for

TABLE II
WEB TEST TRACES CAPTURED.

Trace	Date	Duration	Number of Skype flows
ISP-3	23 Jul 2007	8h	80
ISP-4	22-23 Aug 2007	16h	85

these three parameters and there will be more terms in test Equations (1) and (2). We assume that a bigger number of terms in test Equations produces more accurate results, so this is another reason to filter flows with few requests per page.

From Traces ISP-1 and ISP-2, captured with a two months delay, we may assume that the training dataset generated remains valid for all this time period. There are also some differences between HTTP versions 1.0 and 1.1. We intended to perform one analysis for each HTTP version, but the number of HTTP/1.0 messages was less than 5% of the total messages in all traces. So, all analysis were based only on HTTP/1.1 data and the HTTP/1.0 messages were discarded.

B. Web Test Dataset

In order to evaluate our detection methodology, we captured the evaluation traces shown in Table II. Our collaborative ISP provides valid dynamic IP addresses for its clients, and there are no closed ports or firewall restrictions in their way to the Internet. We suppose there is no Skype traffic in this trace other than our Skype flows, since it only uses Web ports as a fallback mechanism. For all traces, the Skype calls used for evaluation were produced by a small network of computers behind port-restrictive firewalls. It was used Skype versions 1.3 and 1.4 for Linux and version 3.5 for Windows.

V. EXPERIMENTAL RESULTS

In order to present our results, we use ROC curves, a graphical plot of the sensitivity against (1−specificity) of a binary classifier. ROC is an acronym for receiver operating characteristic, sensitivity is the same as true positive rate and (1−specificity) is equal to false positive rate. The classifier has a discrimination threshold that is varied to produce different points in the curve. We have a better classification as the curve approaches the perfect result when all true positives are found (true positive rate is 1) and no false positives are found (false positive rate is 0). The true positive rate is estimated as the number of positive events correctly classified over the number of total positive events and the false positive rate is estimated as the number of negatives incorrectly classified over the number of total negatives.

A. Detecting Skype flows

The empirical distributions shown in Section IV-A are now compared with data generated from each individual flow. In our test traces ISP-3 and ISP-4, we captured TCP/IP headers from ports 80 and 443 and manually identified all flows generated from our Skype calls to serve as reference for the output of our software. At first, each metric was evaluated separately from the others and then we evaluate the case when all of them are jointly considered. All Skype flows generated used port 443 to send traffic.

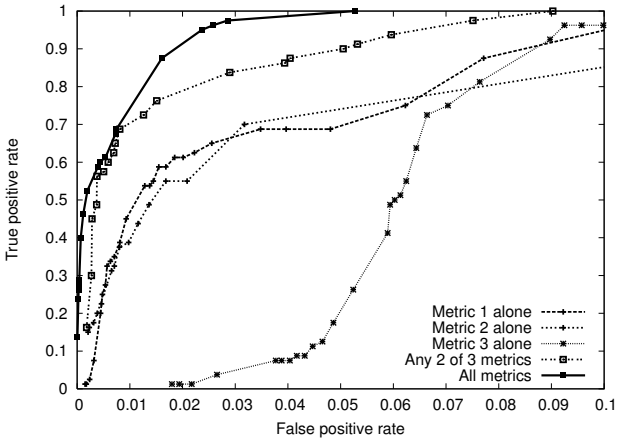

Fig. 4. Evaluating χ^2 ROC Curves.

TABLE III
TRUE POSITIVE RATE (TPR) AND FALSE POSITIVE RATE (FPR) OBTAINED WITH THRESHOLDS 1, 2, 3 FOR TRACE ISP-3 AND χ^2 DETECTION.

Point	Thr. 1	Thr. 2	Thr. 3	TPR	FPR
1	3,000,000	100,000	5,000	0.1375	0.0001
2	3,000,000	50,000	5,000	0.2375	0.0002
3	2,000,000	50,000	5,000	0.2625	0.0003
4	2,000,000	50,000	2,000	0.2750	0.0003
5	1,500,000	50,000	2,000	0.2875	0.0004
6	1,500,000	25,000	1,000	0.4000	0.0007
7	1,500,000	10,000	1,000	0.4625	0.0011
8	1,000,000	10,000	1,000	0.5250	0.0019
9	500,000	8,000	800	0.5875	0.0039
10	500,000	5,000	750	0.6000	0.0043
11	250,000	5,000	750	0.6125	0.0054
12	100,000	5,000	750	0.6750	0.0073
13	100,000	4,000	500	0.6875	0.0075
14	20,000	1,000	500	0.8750	0.0161
15	10,000	1,000	500	0.9500	0.0236
16	10,000	1,000	250	0.9625	0.0258
17	10,000	500	200	0.9750	0.0285
18	5,000	200	100	1.0000	0.0527

We tested some different configurations in the detection program to search for the best results. As shown in Figure 4, each metric used alone has an inferior classification performance in comparison with the combined solution. Metrics 1, 2 and 3 are Web request size, Web response size, and interarrival time between requests, respectively. We also test a detection based only on two metrics, a detection based on any two of the three metrics and a detection based on both tests (χ^2 and K-S). They were all less accurate in comparison with the chi-square test using all three metrics combined. In this case, we have three threshold values for each point. The method used to find optimal points can be summarized as follows: each parameter is individually incremented and decremented by small steps and the best values are selected after the generation and test of a large list of threshold values. In the final sequence, all threshold values must be in decreasing order. In Table III we have the optimal values obtained for trace ISP-3 and χ^2 detection when all thresholds are required for a positive

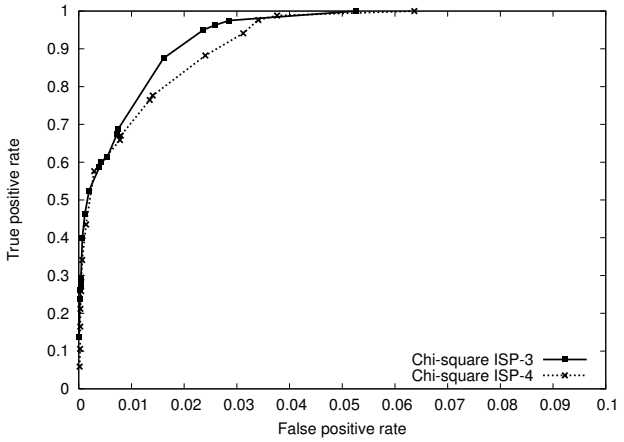

Fig. 5. ROC Curves for χ^2 detection.

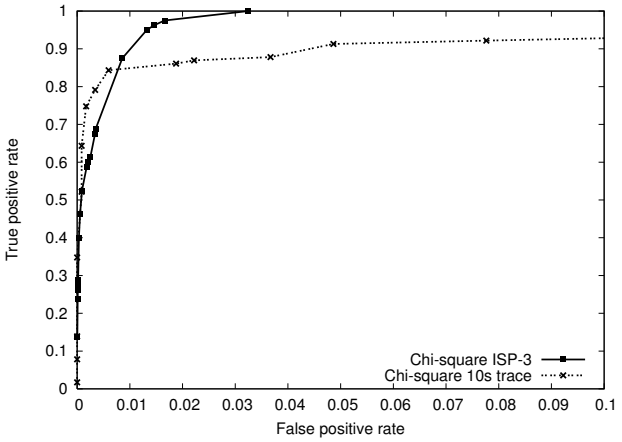

Fig. 7. Comparison between an offline and a time-limited (10s) χ^2 detection.

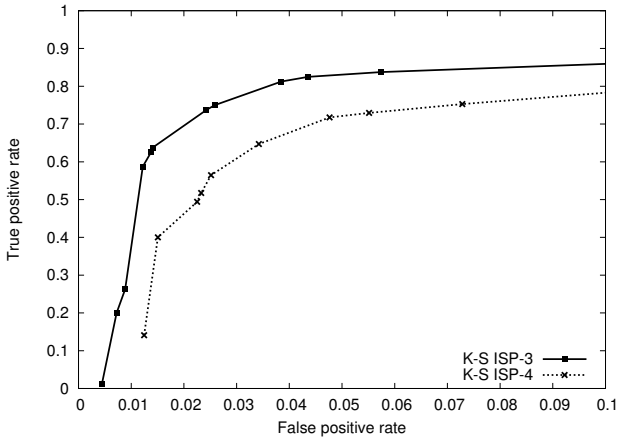

Fig. 6. ROC Curves for Kolmogorov-Smirnov D detection.

classification. Each line represents a point in the curve shown in Figure 4 obtained with all metrics. As the threshold value decrease, the true positive rate grows, but the false positive rate also increases.

The ROC curves for the chi-square detection are shown in Figure 5. The parameters were calculated only for flows with more than 20 requests, performing a total of 17,374 flows in Trace ISP-3 and 24,662 in Trace ISP-4. We observe that both graphs had similar results, but the results for Trace ISP-3 were slightly better. We can achieve around 90% of 80 Skype flows correctly identified (i.e. true positive rate) with less than 2% of 17,294 non-Skype flows incorrectly identified as such (i.e. false positive rate). Likewise, our experimental results also show a 100% detection rate with around 5% of false positives.

The results for the K-S detection are presented in Figure 6. For Trace ISP-3, we can achieve a true positive rate of 70% with a false positive rate around 2% or a 80% detection with 5% of false positives. Comparing Figures 5 and 6, we observe that the K-S results are not so good as the chi-square detection as the points over the χ^2 ROC curve are always closer to the top left corner in comparison with the K-S curve.

Our experimental results thus suggest that the chi-square detection is better than the K-S detection to efficiently detect Skype flows in Web traffic using our methodology. Therefore, considering the experimental results for the chi-square detection shown in Figure 5, our methodology provides enough flexibility for the network management to adopt different approaches regarding the possible detection of Skype flows in Web traffic. At the one hand, if one wants a Skype detection system with few false positive errors (i.e. a conservative approach), one may choose the thresholds used to generate a point near the Y axis at the expense of a reduced detection rate. At the other hand, if one wants to capture almost all true positives (i.e. a loose classification), one may choose the thresholds used in a point near the top axis at the expense of a higher false positive rate.

B. Evaluating real-time detection

The results obtained in previous section were all based on a offline analysis of captured data. However, a network administrator may want to identify the Skype calls that are currently using the network, not the Skype calls made some minutes or hours ago. In this section, we evaluated our algorithm performance taken into account time constraints. The methodology proposed for a real-time detection depends on the same training data but differs in the detection part: this time the data is captured and analyzed using short time intervals. We chose 10 seconds as a reasonable time interval between updates for our detection tool. Skype calls are usually larger than that and network administrators can wait that long for receiving updated information.

For an evaluation, we generated a new test dataset to simulate a real-time detection. We extracted from test traces ISP-3 and ISP-4 a total of 125 capture files with 10 seconds each. These 125 capture files were not contiguous, but separated with various time intervals in order to get different Skype flows in each capture file. After the generation of these files, we manually identified all Skype flows present in them to serve as reference for our detection, counting 115 flows. We launched the χ^2 detection tool used in previous section to generate a new ROC curve. The only modification in our software was a

lower limit for the number of requests present in a flow. Given the small size of capture files, we calculated parameters for every flow with more than 10 requests. The χ^2 ISP-3 detection curve was recalculated after this modification for comparison and the results of our evaluation are shown in Figure 7.

We observe in Figure 7 that the χ^2 detection using the newly generated trace (the set of all 10s capture files) had a true positive rate up to 85% with a smaller number of false positives compared to the χ^2 detection using the ISP-3 trace. Beyond this point, the number of false positives grows significantly, and the χ^2 ISP-3 combination had more accurate results than the χ^2 10s analysis. The time needed to analyse each captured file was insignificant compared to the 10s interval and we used a standard computer for this job. This result suggests that this approach with a 10 seconds time bin can be sufficient for detection, but we can expect some Skype flows to be not distinguishable from Web flows. With a larger time bin, the curve is expected to approach the χ^2 ISP-3 or ISP-4 results, given that the 10s trace is derived from both.

VI. Conclusions

Skype software became very popular in recent years. One of the causes of its success is the high adaptivity provided by the software to operate behind firewalls, proxies or other network elements. Web browsing traffic is a "must-have" service for many institutions and enterprises connected to the Internet. Therefore, it is rather common to find non-HTTP traffic using Web ports to delude firewalls and other network elements. In this paper, we evaluated a Skype detection system based on statistical tests to efficiently detect Skype flows hidden among Web traffic using real-world data gathered at a commercial Internet Service Provider (ISP) and an academic institution. Important features of the proposed Skype detection system include its capacity of detecting Skype traffic without a search for particular Skype patterns or signatures and without regarding payload information. We aim to build a program to detect traffic regardless of Skype version, and difficult to be deluded in newer Skype versions.

Based on a training experimental dataset, we characterize real Web flows to build empirical distributions to represent the "normal" behavior of Web traffic. We manually produced Skype traffic to build our Web evaluation dataset and verify that the proposed metrics are able to identify Skype flows hidden among HTTP traffic. Using two simple Goodness-of-Fit tests, the χ^2 statistic and the Kolmogorov-Smirnov test, we show that Skype flows can be clearly detected, but our results suggests that the χ^2 test is a much better choice.

Our results are dependent on a training dataset but Figures 1, 2 and 3 suggest that the same set of empirical distributions can be used for several weeks and even for other institutions. Figure 7 suggests that our real-time proposal is feasible and it can achieve good results compared to our original methodology. As future work, we intend to further analyze the real-time detection by investigating the minimum time interval needed to achieve 100% of Skype flow detection with a reasonable false positive rate. We also intend to build and evaluate an optimized version of our tool to perform real-time monitoring in network links. The proposed HTTP workload model can be seen as a building block to the development of an automatic detection system of other kind of non-HTTP flows hidden in Web traffic, such as P2P file sharing and media streaming applications. In future work, we plan to investigate this possible generalization of our current Skype detection method and the validity of the training dataset.

Acknowledgment

The authors would like to thank all the support received from IME, LNCC and the collaboration of Antônio Tadeu Gomes, Marcos Gomes Pinto Ferreira and Marcos Vinícius do Couto for capturing Web traffic.

This work was supported in part by CNPq and FAPERJ.

References

[1] IANA, "Port numbers," http://www.iana.org/assignments/port-numbers.

[2] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," in *SIGCOMM'05: Proceedings of the 2005 ACM conference on Applications, technologies, architectures, and protocols for computer communications*, 2005, pp. 229–240.

[3] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy, "Transport layer identification of P2P traffic," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 2004, pp. 121–134.

[4] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 2004, pp. 41–54.

[5] E. P. Freire, A. Ziviani, and R. M. Salles, "On metrics to distinguish skype flows from HTTP traffic," in *LANOMS 2007: Proceedings of the 5th Latin American Network Operations and Management Symposium*, Sep 2007.

[6] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing skype traffic: When randomness plays with you," in *SIGCOMM'07: Proceedings of the 2007 ACM SIGCOMM*, Aug 2007.

[7] P. Barford and D. Plonka, "Characteristics of network traffic flow anomalies," in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, Nov 2001.

[8] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 2004, pp. 201–206.

[9] ——, "Mining anomalies using traffic feature distributions," in *SIGCOMM'05: Proceedings of the 2005 ACM conference on Applications, technologies, architectures, and protocols for computer communications*, 2005, pp. 217–228.

[10] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *IMC '05: Proceedings of the 2005 Internet Measurement Conference*, pp. 331–344.

[11] B. A. Mah, "An empirical model of HTTP network traffic," in *INFOCOM '97: Proceedings of 16th Joint Conference of the IEEE Computer and Communications Societies*, 1997.

[12] H.-K. Choi and J. O. Limb, "A behavioral model of web traffic," in *ICNP '99: Proceedings of the 7th International Conference on Network Protocols*. IEEE Computer Society, 1999, pp. 327–334.

[13] A. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proceedings of the Passive and Active Measurement Workshop (PAM2005)*, March/April 2005.

[14] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, pp. 23–26, 2006.

[15] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, "Unexpected means of protocol inference," in *IMC '06: Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference*, 2006, pp. 313–326.

[16] Y. J. Won, B.-C. Park, H.-T. Ju, M.-S. Kim, and J. W. Hong, "A hybrid approach for accurate application traffic identification," in *4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, Apr 2006, pp. 1–8.

[17] S. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," in *INFOCOM'06: Proceedings of the 25th IEEE International Conference on Computer Communications*, Apr 2006.

[18] S. Guha, N. Daswani, and R. Jain, "An experimental study of the skype peer-to-peer VoIP system," in *IPTPS'06: Proceedings of the 5th International Workshop on Peer-to-Peer Systems*, February 2006, pp. 1–6.

[19] K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley, "Characterizing and detecting relayed traffic: A case study using skype," in *INFOCOM'06: Proceedings of the 25th IEEE International Conference on Computer Communications*, Apr 2006.

[20] S. Ehlert, S. Petgang, T. Magedanz, and D. Sisalem, "Analysis and signature of skype VoIP session traffic," in *CIIT 2006: 4th IASTED International Conference on Communications, Internet, and Information Technology*, Nov/Dec 2006, pp. 83–89.

[21] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Quality and Reliability Engineering International*, vol. 17, no. 2, pp. 105–112, 2001.

[22] J. M. Estévez-Tapiador, P. García-Teodoro, and J. E. Díaz-Verdejo, "Measuring normality in http traffic for anomaly-based intrusion detection," *Computer Networks*, vol. 45, no. 2, pp. 175–193, Jun 2004.

[23] K. Pearson, "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *Philos. Mag. Series 5*, vol. 50, pp. 157–172, 1900.

[24] F. J. Massey, Jr., "The Kolmogorov-Smirnov test of goodness of fit," *Journal of the American Statistical Association*, vol. 46, pp. 68–78, 1951.

[25] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer VoIP calls on the internet," in *CCS'05: Proceedings of the 12th ACM conference on Computer and communications security*, 2005, pp. 81–91.

[26] V. Jacobson, C. Leres, and S. McCanne, "tcpdump," http://www.tcpdump.org/.

[27] J. Elson, "tcpflow," http://www.circlemud.org/~jelson/software/tcpflow/.