Passive Monitoring of HTTPS Service Use

Pierre-Olivier Brissaud, Jerome Francois, Isabelle Chrisment, Thibault Cholez, Olivier Bettan

Passive Monitoring of HTTPS Service Use

Pierre-Olivier Brissaud*†, Jérôme François*, Isabelle Chrisment*,Thibault Cholez* and Olivier Bettan†
*Universite de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France, firstname.lastname@inria.fr
†Thales Service, Palaiseau, France, firstname.lastname@thalesgroup.com

***Abstract*—HTTPS is used today to secure the majority of web communications and so enhance user privacy. Therefore, traffic monitoring techniques must evolve to remain useful, especially to support security considerations, as for example detecting and filtering the forbidden uses of a web service. However, privacy should remain as intact as possible. This paper describes a new passive and transparent method to infer the use of a HTTPS service by extracting and interpreting only meaningful metadata derived from the encrypted traffic without deeply profile individual users. We propose a model using the sizes of objects loaded in the HTTPS service as a signature, by leveraging kernel density estimation, supporting then a classification function. We assess this approach extensively on the *Google Images* Service although our approach remains valid for some other services. We succeed to achieve an accuracy of 99.18% when detecting particular keywords to be searched over a large dataset of 115,500 distinct keywords.**

I. Introduction

According to the report of the Electronic Frontier Foundation [**?**], half of the Internet traffic was protected by HTTPS in February 2017. Although defending and improving user privacy are highly important, monitoring, detecting and mitigating bad behaviors are also legitimate, such as tracking illegal activities undergone through the Internet or inappropriate activities in an enterprise network. However, both of these goals are quite antagonist. Usual monitoring techniques based on simple filtering rules such as port-based or deep packet inspection have become obsolete given the generalization of encrypted web applications.

To keep an accurate view of a traffic within an enterprise network, decryption proxies are usually deployed but they totally break user privacy by being able to decrypt any single HTTPS web page. Actually, the initial objective of monitoring bad behaviors is not to track every request of any user but rather to raise an alert or block users' traffic when they do not respect pre-established rules. We thus provide a solution to allow detecting an inappropriate use of a service (based on static pre-established rules) without revealing the full activity of the users, and without entirely denying access to this service, to assure a good trade-off between security and privacy.

The solution presented in this paper is (1) passive as it only collects encrypted HTTP (HTTPS) traffic, and (2) totally transparent as no specific software is supposed to be installed at the user side. The only pre-requisite is a learning phase to derive signatures based on a specific rule set. More specifically, the particular use of a service leads to the loading of different objects in an HTTPS (HTTP/1.1+TLS) page which exhibit different observable characteristics that can be used to build a signature. In that sense, our approach is complementary to website fingerprinting and service or traffic classification methods. In this paper, we demonstrate the feasibility of such a monitoring scheme on different services and conduct an evaluation on *Google Images*. While traffic classification techniques can identify automatically web and HTTPS traffic, website fingerprinting will reveal that the *Google Images* service is in use. With our in-depth technique we are able to detect a forbidden use of this service, e.g. inappropriate searched contents.

The paper is organized as follows. Section **??** presents related work. Our problem and model are defined in Section **??**. Section **??** gives an overview of the design of the solution and Section **??** and **??** explain the data collection process and the evaluation respectively. Finally, Section **??** discusses the results and Section **??** concludes the paper.

II. Related Work

Previous studies have shown that privacy is not guaranteed by encryption especially with HTTPS. In 1996, Wagner et al. [**?**] highlighted the vulnerabilities of the SSL protocol against traffic analysis. Then, Cheng et al. [**?**] followed by Sun et al. [**?**] and Hintz [**?**] have experimented the feasibility of a website recognition attack using resources' size with closed world assumptions[1]. The publications based on Herrmann et al. [**?**], like Panchenko et al. [**?**], Cai et al. [**?**], Wang et al. [**?**] [**?**] and Hayes and Danezis [**?**] have extended website fingerprinting over anonymization browsers such as TOR [**?**] or JAP [**?**]. They have applied different clustering methods and features, mostly based on packet size distributions, for their fingerprints. Website fingerprinting techniques tend to identify various pages from different websites whereas our goal is to profile users' activities on a single website, i.e. knowing what keywords have been typed, and so, what specific information has been searched.

Analysis of encrypted traffic can be related to different levels as traffic flow classification, service identification or specific application/protocol behavior detection. The first level aims to detect the type of the different flows in the encrypted traffic as surveyed by Velan et al. [**?**]. At a higher level we quote Shbair et al. [**?**] with his framework for service detection over HTTPS. The last level targets specific applications or protocols and detects known behaviors, for example identifying video stream on Netflix by Reed and Kranch [**?**] or VoIP on

[1]Closed world is opposed to open world as it only considers a finite sample for both the training and testing. The open world assumption involves a finite dataset to monitor and unlimited possibilities for the testing phase.

Skype for language or expression detection by Wright et al. [**?**] [**?**]. Those works are built on a precise understanding of the encoding techniques for the video or the audio stream and their aftereffects in the encrypted domain. In [**?**], Coull and Dyer were able to determine the language used in discussions on the IMessage instant text messaging application. They have sorted the type of messages and, based on the text message length, have defined a signature for each language. Song, Wagner and Tian [**?**] have used timing attacks for detecting ssh input and manage to find login password. Finally, IOACTIVE LABS [**?**] have detected the location displayed to the client on Google Maps based on a meticulous learning of the size of satellite images and their contiguity. To identify images on *Google Images*, we propose a similar technique, based on images' sizes and their sequence. However, our method is more general, is robust to small size variations, achieves a direct classification for each loaded page and does not suppose any relationship between consecutive loaded pages.

III. Modeling images searches

A. Problem definition

Many web services as image search engines, online market or social network offer research fields which return pages with many HTTP objects and specifically images. As the images are highly related to the searched keyword, they can be considered as a signature for the keyword, contrary to other web objects (css, etc.) that are used across pages. Thereafter, the encryption should prevent to recognize the keywords and the images if we observe the network traffic. However we demonstrate in this paper that metadata retrieved from an HTTPS connection can lead to detect information on such services.

Considering the previously cited services, they can be used to look after illegal or inappropriate content in a company network. We want to give the network administrator a tool to detect the access to some content without blocking the full service nor being able to discover any other activity that is not explicitly monitored, to respect privacy.

We assume the administrator can thus intercept network traffic and has set a precise keywords[2] list he wants to monitor for each service (the monitored keywords).

When a user requests specific keywords from an image search engine, traffic is generated between the web browser (client) and the server. At this stage, objects and associated metadata are sent to the user. The objective is to know if the traffic generated by the user's request matches some traces produced by a monitored keyword, and possibly to determine this specific keyword.

Assuming L a traffic flow dataset for the learning purpose and M the monitored keyword set, every single element $l \in L$ is a traffic flow corresponding to a single keyword $w \in M$ and defined as a sequence of packets p_i:

$$\forall l \in L, \exists w \in M : l = \langle w, \{p_1, \ldots, p_i, \ldots, p_n\}\rangle \quad (1)$$

[2]In the rest of the paper, keyword refers to the user search which is actually a single or a set of keywords

The aggregation criterion and technique are defined in Section **??**. Based on that knowledge, our model is built as a function f which takes a new sequence of packets p related to a keyword w as input and deduces whether it has been generated or not by a monitored keyword w. If $w \in M$, the function f should recognize p as being in L and return a specific keyword $w \in M$. If not, the function returns u (*i.e.* unknown keyword):

$$\begin{aligned} \forall p &= \{p_1, \ldots, p_i, \ldots, p_n\}, \\ f(p) &= \begin{cases} w \in M \\ u \end{cases} \end{aligned} \quad (2)$$

The problem thus resides in defining f from the learning datasets, *i.e.* all $l \in L$ and the associated keyword $w \in M$.

B. Rationale

In most popular image search engines, several thumbnails are returned for a single keyword. The number of thumbnails is not constant from a keyword research to another, but there are several dozens for each request. Each thumbnail is generated from a single image. Due to image compression techniques with formats like jpeg, the size (in bytes) of images is highly variable and independent of the rendered size (in pixels). As a result, the encrypted size of the thumbnails can vary from one image to another. A keyword is thus associated with numerous encrypted image sizes whose range values span over thousands of bytes. Having two keywords sharing the same set of thumbnails sizes is highly improbable as it is a high combinatorial problem.

C. Refined model

Although in this paper, we study an image search service, the method can be easily transposed to web service which loads numerous and various objects, which can be representative of a particular behavior (*e.g.* searches).

Like the image search service, the content of any web-based application page reflects the service use with various objects loaded depending on the user actions and requests. Therefore, the signatures of a service use can be expressed through some observable characteristics of the objects with some of them being observable even with encryption, especially the size. Indeed, the encrypted size of an object highly depends on its original size. Thus, we refined our model based on Equation (**??**), with s_i the size of each object from a trace[3], as follows:

$$\begin{aligned} \forall l \in L, \exists w \in M : l &= \langle w, \{s_1, \ldots, s_i, \ldots, s_n\}\rangle \\ \forall s &= \{s_1, \ldots, s_i, \ldots, s_n\}, \\ f(s) &= \begin{cases} w \in M \\ u \end{cases} \end{aligned} \quad (3)$$

Actually images are present in many web pages. For example we can mention online shopping website where products are usually represented by image thumbnails as well or social

[3]In the rest of the paper, a trace refer to the packet capture, stored in a pcap file, when requesting a keyword till the full page with all objects is loaded

networks where images are also widely used to illustrate text content and thus images are related with the original request. This is why we select images as significant objects to use for signature construction. Using all objects without distinction would have been possible but introducing some noises due to very generic objects being loaded.

IV. Methodology

A. Overview

As highlighted in Figure **??**, a learning phase is required for the administrator to be able to monitor the user traffic. A database of signatures related to the keywords of the monitored list M is built by the learning module. The list M is provided by the *administrator* (1), the crawler then requests each keyword, a given determined number of times, on the image search engine (2) while the traffic is captured (2'). A profile from all the traces related to a keyword (details in Section **??**) is then derived by the signature generator.

When real users access the search engine (a), the traffic (a') is captured and the part related to the image search engine is extracted. The data are then sent to the classifier (b) which is in charge of determining if a match exists with a signature from the database built during the learning stage (details in Section **??**). Then, the results of the classification engine can be used to raise alerts to the administrator or blacklist a user with bad behaviors (inappropriate searches).

B. Signature generation

As discussed in Section **??** the thumbnail sizes are discriminative of a searched keyword. However, the HTTPS answer containing the thumbnails also includes the HTTP response header. This introduces some noise as the size of the encrypted answer varies (in a scale of few bytes), for the same keyword request. Hence, relying on exact sizes is irrelevant. Thus, the designed method has to catch the variable sizes of the thumbnails and learn automatically this variation when multiple samples are collected.

In this paper, we leverage the Kernel Density Estimation (KDE) technique [**?**] to support our method. Assuming a distribution for the thumbnail sizes of a keyword, KDE estimates the density function. Unlike Gaussian mixture models, KDE is non-parametric in the sense that it does not assume any specific type of underlying probability functions. Hence, the derived density function represents a signature of a keyword and any set of thumbnail sizes can be tested against to verify whether it matches. Let $X = (x_1, ..., x_n)$ be a random distribution, K a kernel function and h the bandwidth parameter of the kernel, the density function is estimated as follows:

$$\begin{cases} \mathbb{R} & \to [0,1] \\ y & \mapsto \frac{1}{nh}\sum_{i=1}^{n} K(\frac{y-x_i}{h}) \end{cases} \tag{4}$$

The bandwidth parameter sets the span of the smoothing of the density function around the provided sample points. The shape of the smoothing is determined by the kernel function, which must fulfil some requirements:

$$\text{Definition } K\colon \mathbb{R} \to \mathbb{R}_+$$

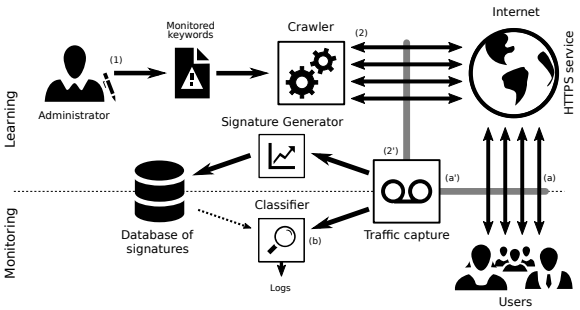

Fig. 1: Global architecture to track HTTPS application use

$$\text{Normalization } \int_{-\infty}^{+\infty} K(y)\,\mathrm{d}y = 1$$
$$\text{Symmetry } \forall y \in \mathbb{R}\colon K(-y) = K(y)$$

In our context, a distribution is composed of all the n thumbnails sizes s_i for a searched keyword k. As we have no prior assumption on data, we use an uniform kernel function defined in Equation (**??**):

$$K(y) = \frac{1}{2} \text{ if } \mid y \mid \leq 1 \text{ else } 0 \tag{5}$$

Assuming Equations (**??**) and (**??**), the signature function noted σ_w is thus defined regarding the distribution of encrypted object sizes (*i.e.* the encrypted thumbnails sizes: s_i):

$$\sigma_w(y) = \frac{1}{nh}\sum_{i=1}^{n}(\frac{1}{2} \text{ if } \mid y - s_i \mid \leq h \text{ else } 0) \tag{6}$$

Our database of signatures D is thus composed of all estimated density functions: $D = \{\sigma_w, w \in M\}$

C. Classification engine

According to Equation (**??**), the classifier processes a list of object sizes and should return a searched keyword w from the list M or u (for "unknown") in case the keyword was not in the monitored keyword list. The whole process is split into three parts: the thumbnails size extraction, the signature scoring and the result filtering.

Considering HTTPS full packet capture pcap traces of a single requested keyword (the dataset collection is explained in Section **??**), extracting an HTTP object size requires first to delimit the latter in the encrypted payload. As no pipelining is available by default on modern web browser with HTTP/1.1 version [**?**], we know that between two client requests, using the same connection, there are only packets corresponding to the HTTP response (header + payload) of the server [**?**], hence only containing a single HTTP object.

Therefore, by analyzing the TCP and TLS headers, a list of sizes for all the encrypted HTTP requests and the corresponding encrypted HTTP response sizes can be built. Because we observe peaks on the requests sizes distribution corresponding to the thumbnail request, we filter the objects sizes to keep only the objects related with thumbnails, that are the most relevant as they are representative of the searched keyword. Thus, we can generate the list of all thumbnails sizes[4] for the

[4]for the rest of the paper we note the thumbnail or image size refer to the encrypted HTTP response size related to the thumbnail

trace subsequent to a keyword query. This list is denoted as $s = \{s_1, \ldots, s_i, \ldots, s_n\}$. Actually, such analysis has to be done for each monitored service.

The scoring consists in computing a single score for each signature (density function) of the database D. For each $\sigma_w \in D$, the score is calculated by evaluating the fitness of each size s_i to the function σ_w and summing over all results :

$$sc_w(s) = \sum_{i=1}^{n} \sigma_w(s_i) \tag{7}$$

A naive classifier would select the keyword associated with the maximum score. However, the classifier may decide that a keyword is unknown as highlighted by u in the classification function in Equation (**??**). The main issue is that there is naturally no signature for u and so a maximum score could not be calculated for an unknown keyword. Actually, the classification engine avoids as much as possible miss-classification and prefers to raise an unknown value if our confidence of the maximal score is too low. To assess the latter, an outlier detection approach is used to check whether this score is much higher than other ones. Assuming the full list of calculated scores L_s, for a given capture, s, of image sizes, the highest score is considered as relevant only if it reaches a threshold calculated from the mean, $mean(L_s)$, and the standard deviation, $std(L_s)$:

$$f(s) = \begin{cases} \underset{w \in M}{\arg\max}\, sc_w(s), sc_w(s) > mean(L_s) + \alpha \times std(L_s) \\ u \; otherwise \end{cases} \tag{8}$$

The filtering parameter α multiplied by the standard deviation defines the minimum deviation from the mean. The impact of this parameter is evaluated in Section **??**.

V. Dataset collection

A. Capturing traces from the Google Images Search Engine

To evaluate our method, we applied it to *Google Images*. For validation purposes, all traces are collected by our own crawler providing both traces for training and testing in the following conditions:

- Client web-browser: Firefox 54.0.1 (64-bit) on debian 8,
- Firefox settings: full screen display with a resolution of 1920×1080 without cache support and HTTP/2 disabled to force HTTP/1.1 (TLS version 1.2),
- English version of *Google Images* (google.com[5]).

Our crawler opens a page at a specific URL and collects the traffic. The packets captured during the page loading of each searched keyword are stored in a pcap file, from which the request and response sizes are extracted for all the loaded objects.

[5]we used https://www.google.com/ncr for disabling regional redirection

B. Dataset constitution based on keywords lists

Our evaluation assumes an open-world situation, *i.e.* the objective is to determine whether a new trace is significantly representative of a monitored keyword (in M) or not. Some of these new testing traces thus correspond to the monitored keywords ($\in M$) while others represent unknown user searches ($\notin M$). A first set of experiments is exclusively based on a restricted dataset $data_1$ of 10,500 keywords from the English Oxford dictionary and allows us to assess our technique and its parameters in various conditions. A second dataset, made from a larger scale experiment, extends the first one with 105,000 other words/expressions from Wikipedia pages titles [**?**] composing $data_2$.

Five traces of each keyword are automatically captured for $data_1$, four are reserved for training, *i.e.* to build meaningful signatures, and one for validation purposes since testing is based on individual traces. $data_2$ traces are only used to verify whether our signatures are robust against new unknown keywords. That is why there is only one trace per each keyword in dataset $data_2$ as training does not use it. All the traffic has been captured during June and July 2017. More detailed information is summarized in Table **??**.

TABLE I: Dataset origins

ID	Origin	Number of keywords	Captures per keyword	Average packets number by trace	Average duration by trace
$data_1$	Dictionary (en)	10,500	5	1,155 pkt/trace	2.94 s/trace
$data_2$	Wikipedia (en) titles pages	105,000	1		

To support reproducible research, a sample of the dataset and a description are available at http://betternet.lhs.inria.fr/datasets/googleimg/, the full dataset is available on demand.

VI. Evaluation

A. Classification metrics

Based on Equation (**??**), a trace p_i with its real label $l_i \in \{L \cup \{u\}\}$ is related to a predicted label $f(p_i)$. Thus we consider $f(p_i)$ as a:

TP (True Positive) if $l_i \neq u$ and $l_i = f(p_i)$, this is the case where the trace p_i corresponds to a monitored keyword from list M and is properly classified accordingly,

FP (False Positive) if $l_i = u$ and $u \neq f(p_i)$, which means that the trace does not correspond to a monitored keyword but is wrongly classified by our system,

WC (Wrong Classification) if $l_i \neq u$ and $l_i \neq f(p_i)$, the trace p_i which is associated with a keyword in M has been properly detected as part of M but the predicted keyword by our system is wrong. Hence, the goal is to evaluate if our method is able to check whether a trace is related to any keyword from M.

We make the distinction between TP and WC because our technique can be tested with two different granularities: with the ability to detect a monitored keyword in the search without distinction among keywords ($TP + WC$) and its ability to exactly identify which keyword has been used (only TP). In

the testing dataset the number of traces related to a monitored or unknown keyword is denoted as $|m|$ and $|uk|$ respectively. Assuming all traces p_i, $|TP|$ is the count of TP traces, $|FP|$ the count of FP and $|WC|$ the count of WC. The following relative metrics are thus computed:

TPR (True Positive Rate): $\frac{|TP|}{|m|}$

FPR (False Positive Rate): $\frac{|FP|}{|uk|}$

WCR (Wrong Classification Rate) The probability that a trace related to a monitored keyword is classified as another monitored keyword: $\frac{|WC|}{|TP|}$

Acc (Accuracy) The ratio of traces which have been well classified independently of their nature (from a monitored or unknown keyword): $\frac{|TP|+|uk|-|FP|}{|m|+|uk|}$

B. Parameters evaluation

In this section, we measure the impact of the different parameters: the bandwidth (h), filtering threshold (α) and the number of monitored keywords ($|M|$) based on extensive experiments. Hence, we use only the small-scale dataset of $data_1$. The objective is to help in configuring our technique and also define the proper parameters for the large-scale experiment presented in the next section.

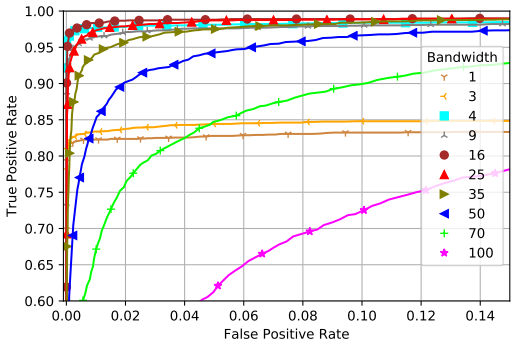

Fig. 2: ROC curve: test bandwidth (small dataset)

In order to determine the effects of the KDE bandwidth parameter, the classifier was trained with 1/10 (1,050) of the keywords and then tested with one trace of all keywords (10,500). This experiment was reproduced three times, each time with a different non-overlapping training keywords set. Average value for the true and false positive rates are shown in Figure **??** where the filtering threshold α varies to draw a single ROC (Receiver Operating Characteristic curve) curve. When the chosen bandwidth (h) increases, the classifier performance is better, with 16 appearing as the best choice (less than 0.2% of false positives with more than 96.8% of true positives) but close to $h = 4$. One exception is for $h = 9$. In that case, a minor performance degradation is shown.

In details, the detection capacity increases from 1 to 4 then between 5 and 16 it first decreases then increases again to

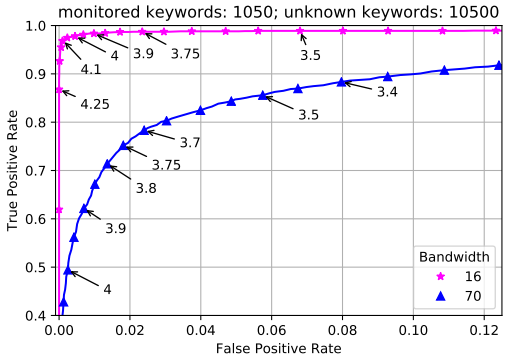

Fig. 3: Impact of h on TPR and FPR assuming a variable threshold α (small dataset)

reach the most efficient value 16. For the bandwidth values greater than 16 the TPR decreases as the bandwidth increases. These variations are related to the variable encrypted size of each thumbnail, i.e. for a same thumbnail, the size can vary of few bytes, which indeed motivates the use of KDE. We experimentally deduce that the variation for the encrypted size is mostly contained in a range of +/- 4 bytes around a central value and few are in a range of +/- 16 bytes.

Secondly, we focus on the filtering threshold (α). Two ROC curves of Figure **??** are selected: with the two different values for the bandwidth $h = 16$ and $h = 70$. They are shown apart in Figure **??** with details regarding the value of the filtering threshold which varies. The higher the filtering threshold is, the lower the TPR and FPR. Logically, the filter dismisses more traces when the threshold increases. It is worth mentioning that the threshold values for the best efficiency depend on the bandwidth because it has a direct impact on the mean score computed for the filtering operation (the density function is smoother when the bandwidth increases).

The last scenario assesses the impact of the number of elements in the list M of monitored keywords. We trained our classifier with 100, 500, 1,000 or 2,000 monitored keywords and use 8,000 unknown keywords for the tests. The bandwidth parameter was set to 16 based on the previous experiment and each experiment was done 3 times with different selected sets of training keywords. Figure **??** highlights that the efficiency of our classifier is inversely proportional to the length of M. Actually, more collisions among keywords can appear when the number of latter increases, and so slightly decreases the efficiency of our technique.

Table **??** summarizes the impact of the bandwidth and filtering threshold parameters on all classification metrics. The WCR almost always equals to 0 except for small filtering thresholds. Hence, when a trace is actually classified, as being part of the monitored keywords list, our technique has a

TABLE II: Result overview(small dataset: $data_1$): 1,050 monitored keyword / 10,500 unknown

h / α	1	4	16	50	1	4	16	50	1	4	16	50	1	4	16	50
	TPR				FPR				Accuracy				WCR			
3.5	0.832	0.983	0.989	0.948	0.0909	0.0788	0.068	0.0602	0.9011	0.9276	0.9379	0.9406	0.015	0.003	0	0
3.9	0.824	0.977	0.983	0.830	0.0182	0.0123	0.0098	0.0083	0.9654	0.9866	0.9895	0.975	0			
4.1	0.819	0.970	**0.968**	0.607	0.0026	0.0018	**0.001**	0.001	0.979	0.9953	**0.9959**	0.9584	**0**			
4.25	0.797	0.941	0.868	0.185	0.0002	0	0	0	0.9789	0.9939	0.9863	0.9155	0			
4.35	0.464	0.223	0.007	0	0	0	0	0	0.9445	0.9194	0.897	0.8963	0			

TABLE III: Result overview (large dataset: $data_1$ & $data_2$): 10,500 monitored keyword / 105,000 unknown

α \ h	1	4	16	50	1	4	16	50	1	4	16	50	1	4	16	50
	TPR				FPR				Accuracy				WCR			
3.5	0.820	0.979	0.986	0.927	0.1042	0.0908	0.0811	0.0739	0.8889	0.9207	0.9250	0.9262	0.017	0.001	0.001	0.001
3.9	0.814	0.973	0.980	0.790	0.0219	0.0201	0.0162	0.013	0.9631	0.9788	0.9834	0.9691	0.004	0	0	0
4.1	0.805	0.965	**0.966**	0.557	0.0052	0.0069	**0.0056**	0.0026	0.9775	0.9884	**0.9918**	0.9574	0.001	0	**0**	0
4.25	0.780	0.932	0.871	0.170	0.0009	0.0038	0.0026	0.0003	0.9792	0.9857	0.9858	0.9242	0			
4.35	0.448	0.253	0.009	0	0	0.0004	0	0	0.9498	0.8762	0.9099	0.9091	0			

high probability to identify the precise keyword which was originally queried. In the best case, the accuracy can reach more than 99% with $TPR = 0.968$ and $FPR = 0.001$ as highlighted in bold in Table **??**.

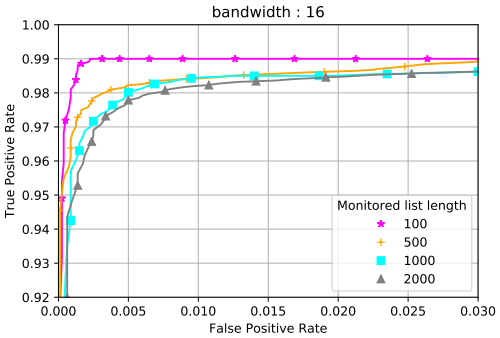

Fig. 4: Detailed values of α (reported on curves) for selected bandwidth parameters (h) (small dataset)

C. Large-scale experiment

Figure **??** reports the experiment of the previous section where 10% of the keyword of $data_1$ are selected to be monitored. Then, we add either unknown keyword for testing or new keywords for learning. By adding traces from non monitored keyword ($data_2$) to be tested (105,000 in total) but still keeping 1050 keywords to be monitored, this proportion is then decreased. As a result (curve 1050/105000), no major difference can be observed, so the signatures remain valid. An update of the signature database is actually necessary when the blacklist M is updated (new keywords added) or for trending keywords which results may change in time.

However, when the amount of monitored keywords increases, in our case with 10,500 keywords (curve 10500/105000), a degradation of the classification performance is shown due to more collisions between signatures. Because the similarities between signatures from different keywords is very low, the degradation is highly limited. Figure **??** shows that the number of TPs is decreased by around 2%. Meantime, the number of FPs is lightly increased by 0.5%. Actually, as reported in Table **??**, the maximum accuracy is 0.9918 with $TPR = 0.966$ and $FPR = 0.0056$. It is obtained with the same parameter configuration as the experiment with the small dataset in the previous section.

VII. Discussion

The stability and the robustness of the signatures are out of the scope of this paper but we share here some observations. The thumbnails returned by *Google Images* can evolve over the time and mostly for keywords related with trending topics. It is possible to continuously generate and capture traffic to update signatures. While many web services rely on user profiles

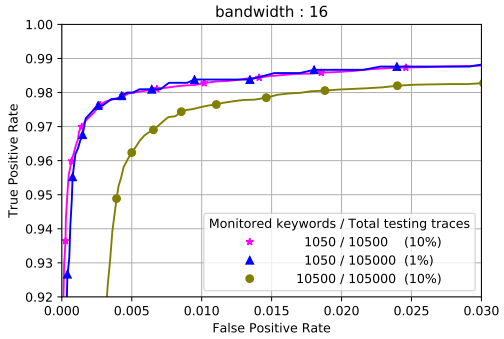

Fig. 5: Roc curve: test on monitored and unknown list length (small/large dataset)

(using cookies for example) to personalize services, we did not observe any evidence that returned thumbnails may be related to the user history, but only to some request options.

The solution is based on the possibility to extract HTTP object sizes and correlate them with their request size. HTTP/2 is therefore challenging because of multiplexing. Indeed, reassembling the packets related to the same response appears no more possible. As a result, the use of the HTTP/2 protocol would prevent the proposed method of this paper. Actually (05/30/2018) 26% of websites have been reported to have migrated to HTTP/2 based on a w3techs report [**?**] and HTTP/1.1 is still today the most widely used HTTP protocol. Furthermore, our technique aims to support the network management, especially monitoring, in an enterprise environment where the administrator may impose the use of a particular version of the protocol in the web browser configuration.

VIII. Conclusion

This paper presented a method to monitor the use of HTTPS services of interest in the context of an enterprise network where security policies must be enforced. Our method relies on the size of traffic generated by individual objects loaded on the web page to build highly discriminative signatures based on KDE. The solution is passive, transparent, as no decryption proxy or middlebox acting as a Man-in-the-Middle is required. It can monitor the use of a predefined list of keywords but not more. Those properties make it better for people privacy than current solutions. The paper presents a deep evaluation when applied to the *Google Image* service. Extensive evaluation was performed based on more than 115,000 real traces. The results show an accuracy higher than 99% with less than 1% of false positives.

In future work, we will focus on testing other methods to evaluate the robustness of HTTP/2 against this kind of fingerprinting.