A Multi-Level Framework to Identify HTTPS Services

Wazen M. Shbair *, Thibault Cholez *, Jerome Francois †, Isabelle Chrisment *
* University of Lorraine, LORIA, UMR 7503, Vandoeuvre-les-Nancy, F-54506, France
† INRIA Nancy Grand Est, 615 rue du Jardin Botanique, 54600 Villers-les-Nancy, France
Email: {shbair.wazen, thibault.cholez, jerome.francois, isabelle.chrisment}@loria.fr

***Abstract*—The development of TLS-based encrypted traffic comes with new challenges related to the management and security analysis of encrypted traffic. There is an essential need for new methods to investigate, with a proper level of identification, the increasing number of HTTPS traffic that may hold security breaches. In fact, although many approaches detect the type of an application (Web, P2P, SSH, etc.) running in secure tunnels, and others identify a couple of specific encrypted web pages through website fingerprinting, this paper proposes a robust technique to precisely identify the services run within HTTPS connections, i.e. to name the services, without relying on specific header fields that can be easily altered. We have defined dedicated features for HTTPS traffic that are used as input for a multi-level identification framework based on machine learning algorithms. Our evaluation based on real traffic shows that we can identify encrypted web services with a high accuracy.**

I. Introduction

The amount of websites encapsulating HTTP in Transport Layer Security (TLS) tunnels has evolved rapidly over the past three years. It accounts, according to French ISPs [1], for 50% of Internet traffic in 2015 against only 5% back in 2012. Most of functionalities previously offered by desktop applications (File Transfer, Office App., etc.) are now run over the cloud and accessed through HTTPS on web browsers or mobile applications at any time and from anywhere [2]. TLS is now one of the key protocols of the Internet. On one side, websites are encouraged to use TLS to protect user privacy and security. On the other side, this security solution comes with issues related to the security management of encrypted traffic. In fact, encryption makes firewalls and Intrusion Detection Systems (IDS) blind to the transferred content and render difficult to enforce security policies to protect companies and individuals from malicious websites hidden within encrypted traffic [3].

Identification of HTTPS traffic is motivated by network administrators who need to gain knowledge about traffic inside their network. In fact, the dependability and security of networks and infrastructures will be lowered if we do not have suitable methods to identify HTTPS traffic. For example, an enterprise's sensitive information may be leaked out over daily used web services or social media that run over encrypted connections. Such leaks are hardly noticed [4]. However, security policies cannot be effectively enforced on encrypted traffic by using current techniques as discussed in [5], because most of them (such as port-based, Deep Packet Inspection (DPI), IP address, DNS or Server Name Indication (SNI) filtering) can be easily bypassed or loose their power against HTTPS traffic. Above all, the existing commercial solutions (FireEye, Forefront [6], etc.) or current IDSs use a controversial approach consisting in the decryption of the traffic in the middle (HTTPS proxy) to analyze it, and by doing so, they deny the right to privacy for network users. Our research question is thus to create privacy-preserving security solutions that allow efficient identification of HTTPS traffic without relying on any decryption.

The related work in the field of encrypted traffic monitoring without decryption can be divided in two main categories. On one side, some studies identify the type of applications (Web, P2P, SSH, VOIP, etc.) behind the encrypted traffic [7]–[9]. On the other side, some studies aim to recognize the accessed web pages of a particular site accessed over secure tunnels, such as SSL-Proxy, SSH-Tunnel, The Onion Routing (Tor) and HTTPS, by using Website Fingerprinting (WF) [10]–[13]. The related work is not satisfactory to address our research question. Identifying the type of encrypted applications is too generic-grain, while the WF is too fine-grained, as it works at the page-level with static content, and is no longer adapted to today's web fetching dynamic content from multiple Content Delivery Networks (CDN). For instance, a user accessing Google Maps will be characterized as HTTPS traffic with the first type of technique whereas the second type will be only able to identify a specific page of the service (mainly the homepage). However, accessing the service may not be from the home page (embedded maps in other websites for example) which thus can be used to bypass the identification. We claim that a service-level identification that can identify the precise services accessed through HTTPS is necessary and would constitute a huge step toward the elaboration of security-solutions that could properly handle encrypted web traffic. In the previous example, our approach will identify when Google maps is used independently of the access method.

Our main contribution is a complete framework to identify accessed HTTPS services in a traffic dump. Our framework includes several innovations increasing the identification accuracy. First, we define a new set of statistical features extracted from the encrypted payload of reassembled TCP connections. Secondly, we propose a multi-level classification approach, where training dataset is processed in a hierarchical fashion based on the domain name: first, the dataset is grouped based on root-domains (for example "google.com") and for each partition, a more fine-grained classifier is used based on sub-domains, where we can differentiate among services (for example "maps.google.com", "drive.google.com").

The rest of the paper is organized as follows. Section II

978-1-5090-0223-8/16/$31.00

provides background on TLS and the SNI extension. Section III presents related work on website fingerprinting and encrypted traffic classification. The proposed HTTPS identification framework is described in Section IV. Section V presents our methodology, feature set and dataset. Section VI evaluates the results of machine learning algorithms applied to our features and combined with the proposed framework. Finally, Section VII concludes the paper.

II. Background On TLS and SNI Extension

A. Overview of TLS

TLS is a cryptographic protocol built to provide a secure connection protecting the security and privacy between two communicating parties. TLS operates below the Application layer and above Transport layer. It is used extensively in applications, as HTTP, FTP, SMTP and VoIP, where security and privacy are needed. In this paper, we focus on HTTP as it is quickly becoming the major application protocol on Internet today and it supports a large variety of services like web-mail, social networks, multimedia streaming, collaborative edition of documents, etc. HTTP Secure (HTTPS) is technically not a protocol by itself, as it simply HTTP used on top of TLS. When a client and server connect over HTTPS, they first complete a TLS handshake as shown in Figure 1, where a negotiation settles a protocol version, cryptographic algorithms, exchanging SSL certificates for authentication and the process to create shared secrets based on public-key cryptography. After completing the handshake, client and server exchange the information through an encrypted link [14].

B. Server Name Indication Extension (SNI)

In the first days of HTTPS, there was a compatibility issue with virtual hosting, when multiple virtual sites are hosted on a single server with each of them having its own SSL certificate. The SNI extension [15] to TLS handshake was proposed to solve mapping between a requested website and the corresponding SSL certificate [16]. SNI holds the destination hostname to which a client is attempting to access at the beginning of a TLS negotiation. So, HTTPS servers can use the SNI to make the mapping between virtual sites and the related certificates. Nowadays, almost all current server software, web browsers, operating systems and certificate authorities support the SNI extension. Moreover, since the SNI is a meaningful and simply extracted value from Client-Hello messages, some firewalls and web-content filtering solutions use the SNI to identify and filter unwanted HTTPS traffic.

In step 1 of Figure 1, a firewall can inspect the SNI within the Client-Hello message to check if the "server name" is allowed or not, so the firewall may reset the connection or allow the Client-Hello message to pass toward the destination server, and further complete the TLS handshake. This filtering based on SNI was evaluated in [5], where a tool, named Escape[1], was built proving that SNI can be easily faked to bypass such firewall systems and still access the filtered websites. Therefore, it is not reasonable to solely depend on SNI for identifying HTTPS traffic in case of attack scenarios, where the attacker will try to hide his activity.

[1]http://madynes.loria.fr/Research/Software

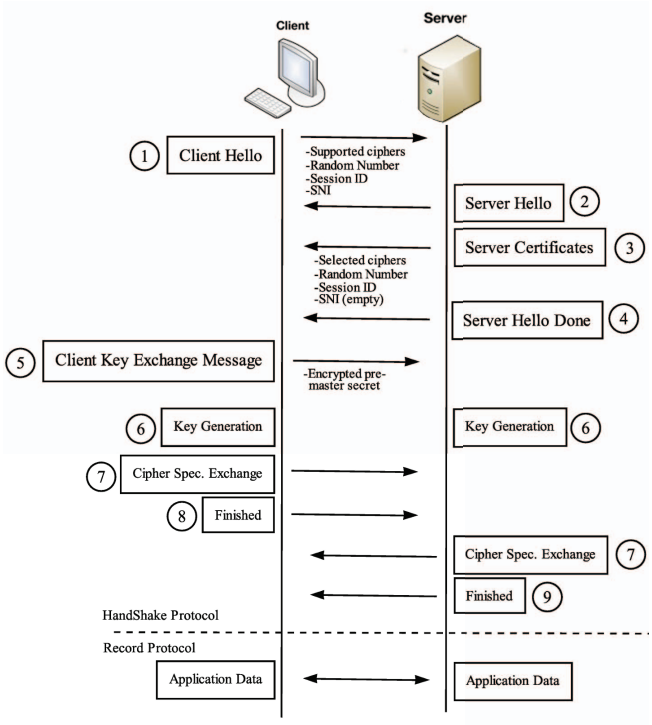


Fig. 1: TLS handshake protocol

III. Related Work

Internet traffic classification techniques have evolved alongside the development of Internet. They range from port-based classification to DPI including flow-based and host-based classification techniques. Previous works done on classification of unencrypted traffic have shown good results, but are suddenly useless against the increasing amount of encrypted traffic, as reviewed in [17]–[19]. Encrypted traffic classification methods face many challenges such as their practical applicability or privacy concern in case of payload decryption (HTTPS proxy). Existing works in the identification of encrypted traffic aim either at identifying the type of applications, such as (Web, Mail, P2P, VoIP, SSH, Skype, etc.) or at identifying accessed pages from a website (Website Fingerprinting).

A. Identifying the type of Applications

The usage of encryption in many network applications creates a real issue linked to the identification of such applications. For that, the flow-based statistical approach has an important advantage related to its applicability because it does not rely on data payload. Bernaille et al. in [7] propose a method based on the size of the first few packets with clustering algorithm using Gaussian Mixture Model. However, the statistics approaches have a lower level of accuracy and computation overhead, for that they have not been widely utilized [17].

Many research efforts have been proposed to overcome the limitations (computation overhead, low accuracy) of flow-based statistics, such as algorithms from different fields like Machine Learning (ML), signal processing, statistical models [18]. Valenti et al. [20] overview ML techniques and show

their benefits in traffic classification problem. In literature work, ML algorithms have been applied for encrypted traffic classification, where they have proven promising results, and this opens the door for the ML application in intrusion detection systems, anomaly detection and traffic monitoring and management [21] [17]. Alshammari et al. [8] use the ML algorithm C4.5 for classifying the encrypted traffic generated from SSH and Skype. In this context, the C4.5 approach performs much better than other ML algorithms. McCarthy et al [22] have investigated the relevance of ML approaches with statistical flow features to identify TLS traffic without using port numbers, IP addresses, or payload information. They use a set of statistical features based on packet length, inter-arrival time and the duration of the flow as input features for ML algorithms. In our work, while reusing part of these features, we propose additional features based on the payload statistics to better identify services run in TLS connections. Schatzmann et al. [9] propose a method to identify Webmail traffic from HTTPS traffic, based on network-level data. The set of features they use depends on Webmail usage behavior such as the periodicity between two AJAX-based client checks for new messages. But the study is limited to mail service, however a wide set of services needs to be studied.

A recent and deep survey in this field [23] identified a couple of limitation in this literature. (1) Identifying the type of encrypted traffic is not enough because the real challenge is to identify the underlying services. We even add that, when considering objectives related to security, the accessed web service should be precisely identified to properly manage the traffic. (2) Most of the literature has been focused solely on SSH while TLS is now by far the most used and the richer protocol in terms of usage.

Our challenges is to precisely name the service that generates HTTPS traffic. The previous works generally consider "HTTPS" as a single class, even if web applications can provide very different kinds of services.

B. Website Fingerprinting (WF)

WF is defined as the process of identifying the URL of web pages that are accessed. Most of early WF techniques focus on analyzing encrypted HTTP traffic over Tor and SSH-tunnel to recognize the accessed web pages. Cheng et al. [24] propose a prototype to identify the web pages visited over HTTPS connection. The prototype is built based on the objects size referenced in the web page. But this technique is useless with the introduction of connection Pipelining and concurrent connection in HTTP1.1 since they prevent the observer from learning the size of individual objects [10]. Liberator et al. [10] propose two systems to infer the source of encrypted HTTP (not HTTPS) covered by SSH-tunnel. The first one is based on the naive Bayes classifier and the second one on Jaccard's coefficient. Both systems relay on packet lengths while discarding timing information. Herrmann et al. [11] propose a multinomial naive-Bayes classifier based on the normalised frequency distribution of IP packet size for accessed HTTP websites over SSH-tunnel. Panchenko et al. [12] focused only on Tor, which is out the scope of this paper. Panchenko also introduces the concepts of *Open-world* and *Close-world* experiments. In *Close* the training and testing of the classifier takes place over predefined set of websites, while *Open* indicates the usage of both interested and unknown websites. Miller et al. [13] proposes an attack to identify the accessed page among 500 pages hosted at the same website based on clustering techniques to identify patterns in traffic. Their experiments are held in a Close-environment, since it is appropriate for HTTPS identification problem, and so are our own experiments.

All the aforementioned studies, as compared in [13], are intended to identify the home-page or internal-pages from a website. But this is too fine-grained as it works at the page-level, specially in the case of identifying services that offer contents to other web pages, such as Akamai. Our work proposes an intermediate method neither too fine-grained nor too generic-grain, by identifying HTTPS at service-level with a flexible and scalable framework.

C. Service Level identification

The method proposed in [25] is the only one that shares our goal of classifying encrypted traffic based at the service-level. The proposed method generates service signatures from TLS payload data and the server IP address. The certificate publication information field in the TLS certificate is then used to label data. However, this method fails when a single certificate is used for multiple services. For instance, it is impossible to differentiate between Google services and to have a fine-grained identification based on the (mutualized) certificate and IP addresses they exhibit. Using the server IP address as an identifier can also be a problem in the case of Virtual-hosting or cloud hosting, where different services can be accessed under a same IP address [5].

IV. An HTTPS Identification Framework

In this section, we present an HTTPS identification framework, which includes ML techniques and a multi-level classification approach. This approach has been used for the first time in the Biology field for classifying the proteins in a hierarchical, tree-like fashion based on shared structural characteristics. First, they group proteins based on their structural class before attempting to assign a protein fold with ML classifier [26]. To our knowledge, it is the first time such an approach has been applied to encrypted traffic classification problem. Most of the existing work have taken a "flat" view toward classification, focusing on identifying the websites and applications directly, while totally ignoring any hierarchical information.

The general idea is building a hierarchical structure, where the top level of the hierarchy is referred to as *Class-level* and each class is itself composed of individual *Folds*. This approach can be applied to the classification of HTTPS services based on the SNI that includes the domain name of the service. The root-domain can refer to *Class-level* and the sub-domain as *Fold*. For example, assume we have HTTPS traffic for Dropbox services such as "photos.dropbox.com" (access hosted photos) and "dl.dropbox.com" (access hosted files) and Google services, such as "maps.google.com" and "drive.google.com". The training and classification hierarchies are built as shown in Figure 2. The *Class-level* classifier is built to differentiate between "google.com" and "dropbox.com". While the *Fold-level* contains two separate classifiers for each class (i.e. one for Google and the second for Dropbox) to

classify between its own services. However, in the case of "flat" view, the classifier needs to distinguish between "photos.dropbox.com", "dl.dropbox.com", "maps.google.com" and "drive.google.com" in one step. Through the rest of paper we will use different terms more adapted to the web context with *Service Provider* referring to the *Class-level*, and the *Service* to the *Fold-level*.

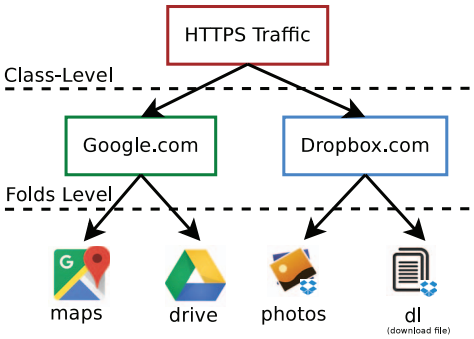


Fig. 2: Multi-level classification approach

Formally, the multi-level classification approach can be described as follows: Assuming $S = \{S_0, .., S_H\}$, where S is a set of service providers, and each element of the set is another set of services $S_i = \{s_i^0, ..., s_i^N\}$, where N represents the number of services belonging to a service provider S_i. For each service s_i^j we defined a set of TLS connections as $T_{i,j,k} = \{t_{i,j,0}, .., t_{i,j,k}\}$, where i is a service provider $0 \leq i \leq |S|$ and j is a service of i, $0 \leq j \leq |S_i|$. Hence our classification approach can be defined as a defining function g where $g(t_{i,j,k}) \rightarrow a, b \quad 0 \leq a \leq |S|, 0 \leq b \leq |S_a|$. The classification can be evaluated as:

$$g(t_{i,j,k}) = \begin{cases} Perfect & i = a, j = b \\ Partial & i = a, j \neq b \\ Invalid & i \neq a, j \neq b \end{cases}$$

Figure 3 gives an overview of our proposed framework for identifying the services running in HTTPS connections. The pre-processing phase starts, as shown in Step 1, with the reconstruction of TLS connections extracted from safe HTTPS traces. Step 2 labels TLS connections thanks to the SNI field and builds the hierarchy between services and service providers. In Step 3, statistical features are calculated. These steps are realized in a control and safe environment to make SNI reliable for the learning stage. The next phase is the training and building of classification models, where the first classification level model is built to differentiate between the service providers as shown in Step 4, while in Step 5 a sub-model for each service provider is built to differentiate among the services that belong to a same provider.

The main rationale behind multi-level approach is to improve the classification model performance by using more precise classification models. The hierarchical representation of the dataset makes the ML algorithm (such as C4.5) more concern about the features that allow a distinction based on more local attributes. As opposed to the flat view classifier, which needs to retrain the whole model if a new service to be added, our approach is more easily extended with either a completely new service provider or a new service. In the first case, we just need to retrain the top-level and add the

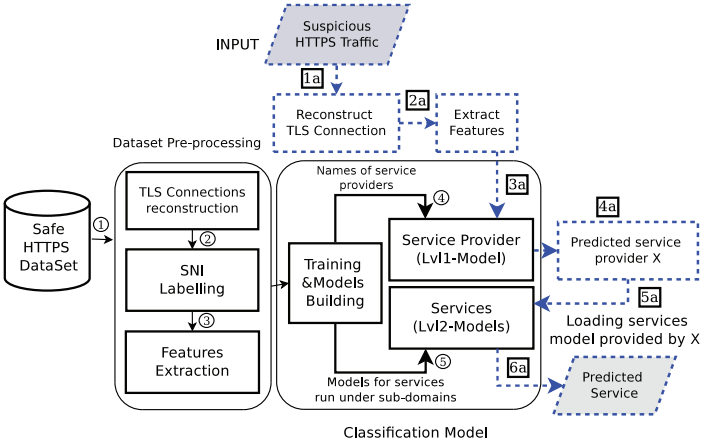


Fig. 3: The workflow of the proposed HTTPS traffic identification framework

related service's classifier in the second level. In the case of a new service from an existing service provider, we need only to rebuild the service provider's model. Figure 3 also illustrates the procedure in a real case scenario when the framework is used to investigate a suspicious HTTPS traffic dump to name the service behind it as follow:

- (1a) TLS connections are reconstructed.
- (2a) Statistical features are extracted.
- (3a) The output of 2a is input for Lvl1-Model classifier
- (4a) The service provider is predicted.
- (5a) A specified model that belongs to the predicted service provider (Lvl2-Models) is loaded to identify the precise service
- (6a) Finally, the name of the predicted service is given.

V. Features and Methodology

In this paper, ML techniques are employed with statistical features over the encrypted payload to identify services run in HTTPS. The common background between ML algorithms is the feature set used to build the model, while each algorithm has his own technique to employ these features to distinguish between classes.

A. The Statistical Features

The proposed framework uses a set of 42 statistical features for a TLS connection. In fact, some of the features, as shown in Table I, are used in [27] and [28] for identifying the type of applications run in TLS connection, but we propose 12 new statistical features related to the encrypted payload, as shown in Table II. These new features are calculated on "Application Data" packets, which contain the overall payload after reassembling TCP segments. The reason is that encrypted payload content holds the most valuable data for client and server, and it gives a closer view of the information exchanged among communication parties. Thus, we only use statistical features over encrypted payload without decrypting the content itself. Moreover, according to [7], the influence of encryption algorithms on the size of packets is negligible with a small increase ranging from 21 to 33 bytes over the original (unencrypted) size for the most common ciphers. In our work,

this result leads us to use the size of encrypted payload without preprocessing or tracking the type of encryption algorithm.

TABLE I: The 30 features from [28]

Client ↔ Server
Total number of packets, Packet size (Average, 25th,50th,75th percentile, Variance, Maximum), Inter Arrival Time (25th,50th,75th percentile)
Client → Server
Total number of packets, Packet size (Average, 25th,50th,75th percentile, Variance, Maximum), Inter Arrival Time (25th,50th,75th percentile)
Server → Client
Total number of packets, Packet size (Average, 25th,50th,75th percentile, Variance, Maximum), Inter Arrival Time (25th,50th,75th percentile)

Statistical features from the payload are used in [29] for high speed real-time classification but were not applied to encrypted traffic. The payload features already provided high accuracy in identifying the main type of application e.g. SSH, FTP, SMTP. Thus we use the benefits of payload statistics to have a fine-grained classification of HTTPS services.

TABLE II: The 12 additional proposed features over the encrypted payload for both directions

Feature name	Directions
Average size	Client→Server, Server→Client
25th percentile size	Client→Server, Server→Client
50th percentile size	Client→Server, Server→Client
75th percentile size	Client→Server, Server→Client
Variance of size	Client→Server, Server→Client
Maximum size	Client→Server, Server→Client

In ML and statistics, feature selection or attribute selection is a process by which we automatically search for a subset of original features that will optimize for higher learning accuracy with lower computational overhead in model construction. The key benefits of this process is reducing over-fitting by removing irrelevant and redundant features, it also improves accuracy and reduces model building time. All these lead to ML-algorithms training and learning faster. Based on the experiments of [30], the Correlation-based Filter Selection (CFS) performs well in terms of classification accuracy and efficiency. Thus it has been used with the Best-First search method to generate a candidate set of features from the features space. The resulting set of selected features is composed of 18 features, listed in Table III. The CFS selects 10 features from our proposed set out of 12 and 8 features out of 30 from the classical ones. This validates the rationale of the proposed features for identifying HTTPS services.

TABLE III: The 18 selected features

Client ↔ Server
Inter Arrival Time (75th percentile)
Client → Server
Packet size (75th percentile, Maximum), Inter Arrival Time (75th percentile), Encrypted Payload(Mean, 25th, 50th percentile, Variance, maximum)
Server → Client
Packet size (50th percentile, Maximum), Inter Arrival Time (25th, 75th percentile), Encrypted payload(25th, 50th, 75th percentile, variance, maximum)

B. Machine Learning Algorithms

In literature work, different ML algorithms have been used for traffic classification. In this subsection, we expose the results of a preliminary experiment conducted with full feature set over the collected dataset for selecting the most promising supervised learning algorithms. The selected algorithms analysed later in conjunction with the proposed features and the framework. As shown in Table IV two algorithms have established a higher accuracy: C4.5 and RandomForest, and are used in the rest of this paper. The *C4.5 Decision Tree* is a hierarchical data structure where all features are used for building internal decision nodes and terminal leaves [31]. The *Random Forests* is a structure consisting of many tree-structured classifiers. Each tree is constructed with random selection of features. The main principle is training several tree classifiers and after the forest is formed. A new object that needs to be classified is given to each of the tree in the forest, each tree gives a class and the forest chooses the class having the most votes (i.e Hard decision) [32]. The Soft prediction is later employed to assess the *Confidence Score* of the framework. In the case of RandomForest algorithm, *Confidence Score* shows the level of agreement between the decision trees. Thus, the final decision is taken based on the maximum sum of predication probabilities, which gives an indication of the validity of the decision [33].

TABLE IV: Preliminary evaluation of ML algorithms

Algorithm	NaiveBayes	RandomTree	C4.5	RandomForest
Max-accuracy	57.9%	85%	**87.8%**	**89%**

C. Dataset Collection and Labelling

An HTTPS dataset has been built, because we needed traces with full TLS payload but no public dataset exists. We tried to involve several users, many services over long time period to make it representative. Building dataset is a research question on its own but, from the aforementioned survey [23], we are in line with similar works. The training dataset has been built in well controlled environment (our lab) with voluntary users, such as students and researchers of our research team. The HTTPS traces of complete user's sessions have been collected. We use the SNI extension for labelling each HTTPS connection, since it directly refers to the specific HTTPS service that is accessed. In our case, we assume that the training traces come from a safe environment where no forged/fake SNI is present during the learning phase. SNI is used as *Ground Truth* for the classification experiments.

The server-name field inside the SNI is detailed enough to exhibit the service name. In some cases, a website page needs to make multiple concurrent requests with servers to render the page's content and store their contents over multiple servers that can exhibit different names in the SNI field. Table V shows some names related to the Google Maps service, when the website makes concurrent connections to render the maps pages. It can be seen that some names show a common prefix followed by a numbers. To avoid over-differentiation of services and ease the learning process, we perform a pre-processing of SNI values by removing the numbers and special characters (like dashes) in order to keep meaningful names.

D. Evaluation Metrics

The performance of our classifier was evaluated by a K-fold cross-validation. In K-fold cross-validation, the dataset is randomly split into K roughly equal parts. The algorithm is

TABLE V: Google Maps servers names

maps.google.com	mt0.google.com	mt1.google.com
khm.google.com	khm0.google.com	khm1.google.com
khmdb0.google.com	khmdb1.google.com	maps.gstatic.com

trained and tested K times, for each $k = \{1, 2, 3, ..K\}$ the algorithm is trained on $K-1$ parts and tested on the kth part. The most commonly used metrics to measure the effectiveness of the ML-algorithms are *Precision*, *Recall*, and *F-Measure*. The *precision* is defined as the proportion of the instances which truly have class A divided by total classified as class A. The *Recall* is similar to True Positive Rate, it measures the precision of the ML-algorithm for a specified class. The *F-Measure* is calculated by precision and recall as shown below.

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

We also use Receiver Operating Characteristics (ROC) graphs for visualizing the performance of our classifier for identifying the right HTTPS service. The simple classification accuracy is often insufficient for measuring a classifier performance [33]. ROC graphs are two-dimensional graphs in which True Positive Rate(TPR) is plotted on the Y axis and False Positive Rate (FPR) is plotted on the X axis at various classifier threshold settings.

VI. Evaluation

This section evaluates the effectiveness and the accuracy of our solution to identify services running in HTTPS connections. Firstly, we provide a statistical overview for the collected dataset. Secondly, our proposed features are evaluated with the "flat" view (i.e. the traditional way of classification). Finally, we present the improvement achieved by the proposed multi-level HTTPS identification framework.

A. Overview of HTTPS Traces Dataset

The collected HTTPS dataset contains more than 288,901 HTTPS connections for different services accessed by volunteer users. Table VI shows the connections number of top services appearing in our collection. Some services are used very frequently, while others are rarely used leading to a small number of connections. However, ML algorithms need a significant training set and we must pre-process the dataset to determine a reasonable threshold for the minimum number of labelled connections per service (i.e. the sufficient number of solved examples for training phase). In Figure 4 we show the relation between this number and the total number of different services we can process. For example, the maximum number of services is when considering all services having at least 5-connections in the traces, while with a minimum of 50-connections, 263 services can be studied (from 107 distinct service providers). In the rest of the paper, we will evaluate the classifier accuracy according to three thresholds (10, 40 and 100 minimum connections per service).

Figure 5 shows how the overall number of training-connections varies when increasing the minimum number of connections per service. Even with a number set to 100, the number of abandoned connections is less than 2% of the overall dataset, which means that 98% of the collected traffic is still used for the classification experiments.

TABLE VI: The main services which have been collected

Service Provider	Number of Connections	Number of Services
Uni-lorraine.fr	71,595	15
Google.com	47,732	29
akamihd.net	15,700	6
Googlevideo.com	4,580	1
Twitter.com	3,325	3
Youtube.com	3,160	1
Facebook.com	3,147	4
Yahoo.com	1,966	19
Cloudfront.com	773	1

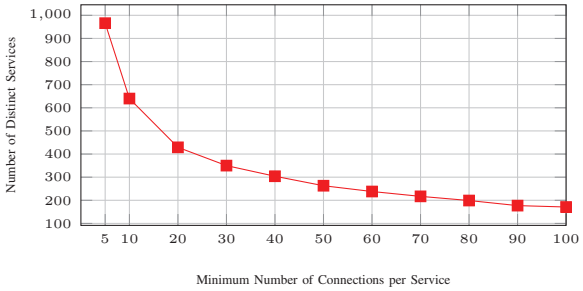


Fig. 4: Overall number of distinct services being given a minimum number of related connections in the traces

B. Feature Sets Evaluation

The classical features from the state-of-the-art, as described in Table I, are used as the baseline to evaluate our features. Tables VII and VIII show the evaluation of the C4.5 and the RandomForest algorithms with different minimum number of connections per HTTPS service. The Weka library was used for this evaluation, with default Weka's configuration to C4.5, while the number of estimators for the RandomForest is tuned to 10. Employing the 10-Fold validation for the C4.5 algorithm achieves an average percent of 83.4%±1.0 precision, as shown in Table VII, while the RandomForest achieves 85.7%±0.4 precision. These tables also highlight the relation between the overall classification accuracy and the minimum number of connections needed: the accuracy is almost stable with 40 connections or more.

TABLE VII: Classical features with C4.5 and RandomForest

#Conn.	C4.5			RandomForest		
-	Precision	Recall	F-Measure	Precision	Recall	F-Measure
10	81.8%	82%	81.7 %	NA%	NA%	NA%
40	83.3%	83.3%	83.1 %	84.9%	85.5%	84.8%
100	84.4%	84.7%	84.4 %	86.4%	86.8%	86.3%

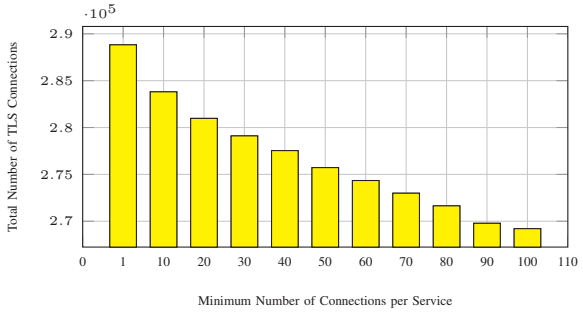


Fig. 5: Overall number of connections in function of the minimum number of connections seen for each service

By extending the classical features with our proposed features (listed in Table II), the overall accuracy increases by 3 points as shown in Table VIII. The C4.5 algorithm now achieves 86.65%±0.7 precision, while the RandomForest achieves 87.82%±0.68 precision. The reduced set of selected features also achieves better results as shown in Table IX: C4.5 achieves 85.87%±0.64 precision and RandomForest 87.60%±0.10 precision.

TABLE VIII: Full features with C4.5 and RandomForest

#Conn.	C4.5			RandomForest		
-	Precision	Recall	F-Measure	Precision	Recall	F-Measure
10	85.4%	85.4%	85.2 %	86.6%	87.2%	86.6%
40	86.4%	86.4%	86.2 %	87.6%	87.9%	87.6%
100	87.6%	87.7%	87.5 %	88.8%	89%	88.7%

TABLE IX: Selected features with C4.5 and RandomForest

#Conn.	C4.5			RandomForest		
-	Precision	Recall	F-Measure	Precision	Recall	F-Measure
10	84.7%	84.6%	84.6 %	86.6%	87.2%	86.6%
40	85.6%	85.7%	85.5 %	87.2%	87.6%	87.1%
100	86.8%	87%	86.8 %	88%	88.3%	88%

Figures 6(a) and 6(b) show a comparison between the two sets of features. It can be noticed that the full set (42 features) achieves higher accuracy compared with classical one, both in C4.5 and RandomForest algorithms. The ROC plots in 6(c), 6(d) visualize the performance of C4.5 and RandomForest classifier with different feature set. Also it shows how the proposed features over the encrypted payload improve the overall performance of the classifiers.

Based on the aforementioned results, the RandomForest algorithm with the full feature set performs the best. When applying our evaluation method described in Section 4, the evaluation gives 90.95% of perfect identification of HTTPS services and 4.5% partial identification (i.e. the service provider is well identified but not the service itself). As a result, in our efforts to reduce the partial identification and improve the perfect recognition of services run in HTTPS, a multi-level classification approach was added to our HTTPS identification framework and is evaluated in the next section.

C. HTTPS Identification Framework Evaluation

As shown in Figure 3 the core component of the proposed framework is the first level model, which predicts the service provider of the HTTPS traffic. Based on the previous section best results, we use RandomForest with 100 connections as minimum number per service to evaluate the framework. We start by evaluating each level to measure the performance of each framework's components and then we evaluate the whole framework as one black box. For the evaluation of the first level model, we still consider all feature set to show the improvement of framework's classifier of each of them. Table X shows that the full feature set achieve a high level of accuracy for identifying the service provider of HTTPS traffic. The ROC analysis in Figure 7 shows that it also improves the identification accuracy of the service provider over the classical set.

In the second level of classification, separate classification models are built for each service provider. Each model has

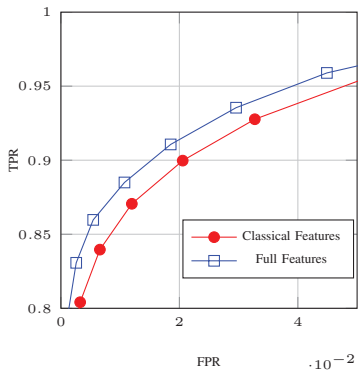


Fig. 7: ROC analysis for the classification model at the top level of multi-level identification

TABLE X: First level model evaluation with RandomForest

-	Precision	Recall	F-Measure
Classical Features	91.9%	92%	91.7%
Full Features	93.6%	93.7%	93.5%
Selected Features	92.6%	92.8%	92.6%

been evaluated separately with the same approach used in the first-level. Table XI summarise the distribution of the overall accuracy for these models. In our dataset, we have 68 distinct service providers that exhibit a service counting more than 100 connections in the traces. 51 service providers have more than 95% of good classification of their own different services. As expected, this result supports the idea that identifying between services from the same service provider with a specific model achieves higher accuracy. For example, the result shows that we can classify among the 19 different Google services, running under "google.com", with more than 93% of perfect identification. Moreover we can notice that the sets of full and selected features perform the same and better than the classical feature set.

The whole framework (Level1&2) has been evaluated by 10-Fold cross validation. The overall accuracy is calculated based on our evaluation method described in Section IV. The results show that we achieve 93.10% of perfect identification and 2.9% of partial identification. To assess the confidence level of the framework, we divide the confidence score range

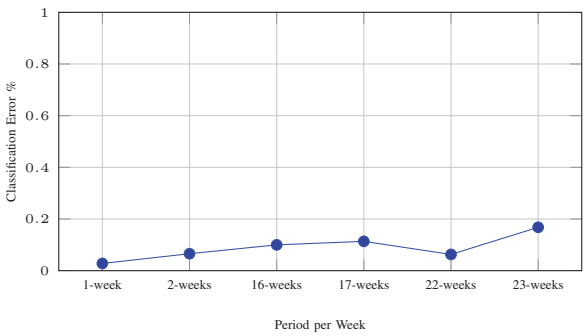
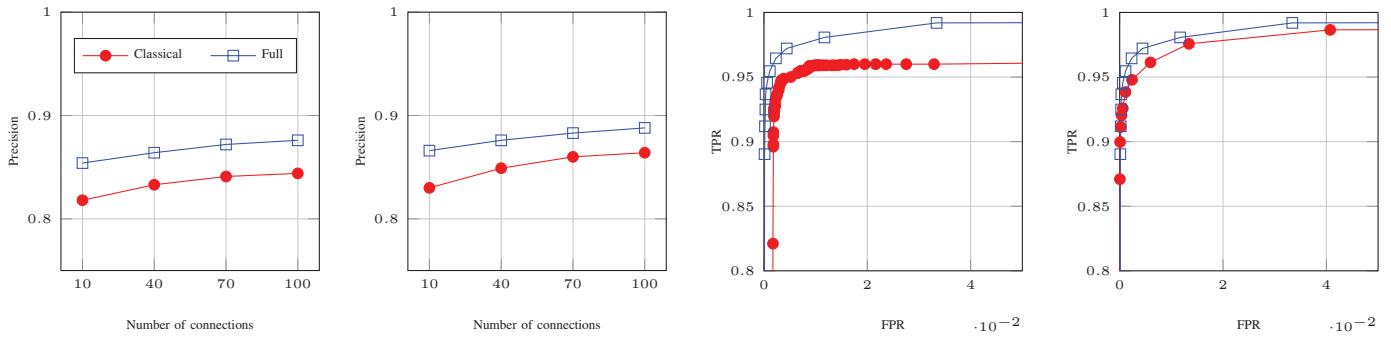


Fig. 8: Effect upon classification error over time



(a) Precision comparison between Classical and Full features with C4.5

(b) Precision comparison between Classical and Full features with RandomForest

(c) ROC analysis for classic and full features with C4.5 Classifier

(d) ROC analysis for classic and full features with RandomForest Classifier

Fig. 6: Evaluation of the proposed features

TABLE XI: The second level models accuracy with 100 connections as minimum number of connections per service

Accuracy Range	Number of service providers		
-	Classical Features	Full Features	Selected Features
100-95%	50	51	51
95-90%	5	5	5
90-80%	6	6	6
Less than 80%	7	6	6

of value [0-1] to 11 sub-ranges. The predictions scores are counted for each sub-range as shown in Figure 9, correct prediction at the top and wrong prediction at the bottom. We can observe that, 86.68% of the predictions are in the sub-ranges [0.8-0.9[, [0.9,1[and 1. Hence, we are almost sure that we identify correctly and so assume the results as relevant. The remaining 13.16% of the predictions is more balanced (46% right and 53% wrong) and will need more analysis, such as DNS or IP address investigation.

Figure 8 depicts the classification errors over time. We can notice that even after 23 weeks without new learning phase, we still identify 80% (error <20%) of HTTPS services. As a conclusion, we can say that the innovations (Multi-level classification, SNI-labelling, new set of features) that we included in the proposed framework were all beneficial to address the difficult problem of the identification of services run in HTTPS traffic. The high level of accuracy finally achieved should allow the framework to be tested in the case of real network.

VII. Conclusion

The encrypted TLS-based traffic comes with new challenges related to the management and security analysis of encrypted traffic. Especially, there is an essential need for new methods investigating the increasing number of HTTPS traffic flows while respecting privacy (no decryption). Most of researches in the literature focus either on identifying the main types of applications or specific encrypted web pages protected by SSH or Tor. But so far, few studies tried to analyse HTTPS websites that can cover very different services. To the best of our knowledge, we are the first to propose such framework, which is primordial as applications tend to be more and more web-based. Therefore our main contribution is a complete framework to identify what are the HTTPS services accessed in

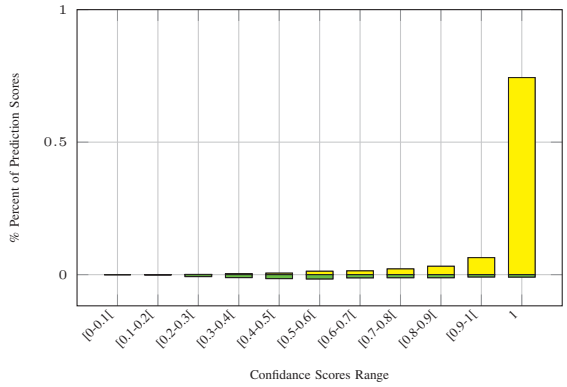


Fig. 9: The confidence scores hits for the multi-level identification framework

a traffic dump, for example to identify communications toward forbidden services in the context of a forensic analysis (i.e. after a revealed security issue). While the SNI field can be used to identify benign traffic, it can be easily forged by a malicious party to hide its activity. To cope with this issue, our framework only uses SNI to label the HTTPS traces of the training set. We also included in the framework several innovations making the identification accurate and the framework scalable. The first is a new set of statistical features extracted from the encrypted payload of reconstructed HTTPS connections. The second is a multi-level classification technique, where in the top level the main service provider is identified, while in the second level the precise services are recognized. We defined an evaluation method and we obtained a high level of accuracy of 93.10%. Based on these good results, our future work will be to adapt and extend our current framework to enable real-time analysis and identification of HTTPS services, with the goal to improve the global security of networks with a new generation of HTTPS firewall.

Acknowledgement

This work was partially funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its FP7 Programme, and by HuMa, a project funded by Bpifrance and Region Lorraine under the FUI 19 framework.

REFERENCES

[1] "Report 2015-0832 from the French regulatory authority for telecommunications (ARCEP). Structure de l'usage de la bande passante des reseaux d'acces a internet sur le territoire francais," http://www.arcep.afr/uplods/tx_gsavis/15-0832.pdf, [In French], Accessed: 07/07/2015.

[2] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The cost of the S in HTTPS," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. ACM, 2014, pp. 133–140.

[3] D. Kolton, A. Stav, A. Wexler, A. E. Frydman, and Y. Zahavi, "System to enable detecting attacks within encrypted traffic," Feb. 22 2011, uS Patent 7,895,652.

[4] "Managing encrypted traffic with Blue Coat solutions," https://www.bluecoat.com/solutions/managing-ssl-and-https-traffic, Accessed: 15/9/2015.

[5] W. M. Shbair, T. Cholez, A. Goichot, and I. Chrisment, "Efficiently bypassing SNI-based HTTPS filtering," in *Integrated Network Management(IM),2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 990–995.

[6] "Configuring HTTPS inspection with forefront threat management gateway (TMG)," http://www.isaserver.org/articles-tutorials/configuration-general/Configuring-HTTPS-Inspection-Forefront-Threat-Management-Gateway-TMG-2010.html, Accessed: 4/1/2016.

[7] L. Bernaille and R. Teixeira, "Early recognition of encrypted applications," in *Passive and Active Network Measurement*. Springer, 2007, pp. 165–175.

[8] R. Alshammari *et al.*, "Machine learning based encrypted traffic classification: identifying SSH and Skype," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. IEEE, 2009, pp. 1–8.

[9] D. Schatzmann, W. Mühlbauer, T. Spyropoulos, and X. Dimitropoulos, "Digging into https: flow-based classification of webmail traffic," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 322–327.

[10] M. Liberatore and B. N. Levine, "Inferring the source of encrypted HTTP connections," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 255–263.

[11] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 31–42.

[12] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. ACM, 2011, pp. 103–114.

[13] B. Miller, L. Huang, A. D. Joseph, and J. D. Tygar, "I know why you went to the clinic: Risks and realization of HTTPS traffic analysis," in *Privacy Enhancing Technologies*. Springer, 2014, pp. 143–163.

[14] T. Dierks, "The transport layer security (TLS) protocol version 1.2 (RFC 5246)," 2008.

[15] D. Eastlake, "Transport layer security (TLS) extensions: Extension definitions (RFC 6066)," 2011.

[16] L. Völker, M. Noe, O. P. Waldhorst, C. Werle, and C. Sorge, "Can internet users protect themselves? challenges and techniques of automated protection of HTTP communication," *Computer communications*, vol. 34, no. 3, pp. 457–467, 2011.

[17] Z. Cao, S. Cao, G. Xiong, and L. Guo, "Progress in study of encrypted traffic classification," in *Trustworthy Computing and Services*. Springer, 2013, pp. 78–86.

[18] W. de Donato, A. Pescape, and A. Dainotti, "Traffic identification engine: an open platform for traffic classification," *Network, IEEE*, vol. 28, no. 2, pp. 56–64, 2014.

[19] C. Bacquet, M. Heywood *et al.*, "Genetic optimization and hierarchical clustering applied to encrypted traffic identification," in *Computational Intelligence in Cyber Security(CICS), 2011 IEEE Symposium on*. IEEE, 2011, pp. 194–201.

[20] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia, "Reviewing traffic classification," in *Data Traffic Monitoring and Analysis*. Springer, 2013, pp. 123–147.

[21] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 4, pp. 56–76, 2008.

[22] C. McCarthy *et al.*, "An investigation on identifying SSL traffic," in *Computational Intelligence for Security and Defense Applications (CISDA), 2011 IEEE Symposium on*. IEEE, 2011, pp. 115–122.

[23] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

[24] H. Cheng and R. Avnur, "Traffic analysis of SSL encrypted web browsing," *URL citeseer. ist. psu. edu/656522. html*, 1998.

[25] S.-M. Kim, Y.-H. Goo, M.-S. Kim, S.-G. Choi, and M.-J. Choi, "A method for service identification of SSL/TLS encrypted traffic with the relation of session ID and Server IP," in *Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific*. IEEE, 2015, pp. 487–490.

[26] K. Marsolo, S. Parthasarathy, and C. Ding, "A multi-level approach to SCOP fold recognition," in *Bioinformatics and Bioengineering, BIBE 2005. Fifth IEEE Symposium on*. IEEE, 2005, pp. 57–64.

[27] Y. Okada, S. Ata, N. Nakamura, Y. Nakahira, and I. Oka, "Comparisons of machine learning algorithms for application identification of encrypted traffic," in *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, vol. 2. IEEE, 2011, pp. 358–361.

[28] Y. Kumano, S. Ata, N. Nakamura, Y. Nakahira, and I. Oka, "Towards real-time processing for application identification of encrypted traffic," in *Computing, Networking and Communications (ICNC), 2014 International Conference on*. IEEE, 2014, pp. 136–140.

[29] F. Dehghani, N. Movahhedinia, M. R. Khayyambashi, and S. Kianian, "Real-time traffic classification based on statistical and payload content features," in *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*. IEEE, 2010, pp. 1–4.

[30] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *Proceedings of the 2008 ACM CoNEXT conference*. ACM, 2008, p. 11.

[31] R. Alshammari and A. N. Zincir-Heywood, "Can encrypted traffic be identified without port numbers, ip addresses and payload inspection?" *Computer networks*, vol. 55, no. 6, pp. 1326–1350, 2011.

[32] J. Zhang and M. Zulkernine, "Network intrusion detection using random forests," in *Third Annual Conference on Privacy, Security and Trust, Proceedings*, October 12-14, 2005.

[33] S. Marchal, J. François, R. State, and T. Engel, "Phishscore: Hacking phishers' minds," in *Network and Service Management (CNSM), 2014 10th International Conference on*. IEEE, 2014, pp. 46–54.