Analyzing HTTPS Encrypted Traffic to Identify User’s Operating System, Browser and Application

Jonathan Muehlstein*, Yehonatan Zion†, Maor Bahumi‡, Itay Kirshenboim∗†
Ran Dubin‡, Amit Dvir*, Ofir Pele†‡

* Center for Cyber Technologies, Department of Computer Science, Ariel University
† Center for Cyber Technologies, Department of Electrical and Electronics Engineering, Ariel University
‡ Department of Communication Systems Engineering, Ben-Gurion University of the Negev

Abstract—Desktops and laptops can be maliciously exploited to violate privacy. There are two main types of attack scenarios: active and passive. In this paper, we consider the passive scenario where the adversary does not interact actively with the device, but he is able to eavesdrop on the network traffic of the device from the network side. Most of the internet traffic is encrypted and thus passive attacks are challenging. In this paper, we show that an external attacker can identify the operating system, browser and application of HTTP encrypted traffic (HTTPS). To the best of our knowledge, this is the first work that shows this. We provide a large data set of more than 20000 examples for this task. Additionally, we suggest new features for this task. We run a through a set of experiments, which shows that our classification accuracy is 96.06%.

Index Terms—Encrypted Traffic, HTTPS, Operating System, Browser, Application

I. INTRODUCTION

There are two main types of attack scenarios: active and passive. Active adversaries try to physically or remotely control the user device. Passive adversaries may violate the privacy of the user by sniffing the network traffic of the devices from the network side. In this work we consider passive attacks.

If network traffic is not encrypted, the task of a passive attacker is simple: he can analyze the payload and read the content of each packet. User activities tracking on the web was proposed in [1]–[3]. This has been done by analyzing unencrypted HTTP requests and responses. A passive adversary may use this information for understanding user actions and revealing information regarding personal interests and habits.

However, most of the internet traffic today is encrypted. This happens both as users start to gain more familiarity with privacy threats and as Google encourages all website owners to switch from HTTP to HTTPS by taking into account whether sites use secure, encrypted connections as a signal in their ranking algorithms [4].

Many works have shown that encryption is not sufficient to protect confidentiality [5]–[39]. Bujlow et al. [27] presented a survey about popular DPI tools for traffic classification. Moore et al. [33] used a Naive Bayes classifier which is a supervised machine learning approach to classify internet traffic applications. Williams et al. [35] conducted a comparison of five machine learning algorithms that were used to classify Internet traffic applications. Auld et al. [34] proposed to use a supervised Bayesian neural network to classify internet traffic applications. Alshamari et al. [36] compared AdaBoost, Support Vector Machines, Naive Bayes, RIPPER and C4.5 in order to classify Skype traffic. Donato et al. [39] presented a method for application classification called the Traffic Identification Engine.

Niemczyk et al. [38] suggested to divide the session to time buckets (10 seconds). The features that were used for each bucket are packet size counts and the time differences between packets. They found the recognition rate of Skype was almost perfect. However, their method was not able to differentiate between browsers and between joint application and browser usage.

Feature extraction methods for traffic classification include session duration [36], number of packets in a session [32], [40], minimum, maximum and average values of inter-arrival packets time [32], [36], payload size information [32], bit rate [41], [42], round-trip time [41], packet direction [43], and server sent bit-rate [44] that has the advantage of overcoming communication problems such as packet loss and retransmissions.

Liberatore and Levine [45] showed the effectiveness of two traffic analysis techniques for the identification of encrypted HTTP streams. One is based on a na¨ıve Bayes classifier and one on the Jaccards coefficient similarity measure. They also proposed several methods for actively countering the techniques, finding these methods effective, albeit at the cost of a significant increase in the size of the traffic stream. Panchenko et al. [46] showed that a Support Vector Machine (SVM) classifier is able to correctly identify web pages, even when the user used both encryption and anonymization networks such as Tor [47]. Cai et al. [48] presented a web page fingerprinting attack and showed that it is able to overcome defenses such as the application-level defenses HTTP [49] and randomized pipelining over Tor.

Matousek et al. [50] presented a technique for the identification of the operating system based on TCP parameters and traffic fingerprinting. Husak et al. [51] proposed real-time lightweight identification of HTTPS clients based on network monitoring and SSL/TLS fingerprinting. The fingerprinting is based on pairing HTTP traffic with SSL parameters of encrypted HTTP traffic. In these works, the systems have to identify the SSL parameters and are not robust to changes in the SSL parameters such as cipher suite.
Exploiting traffic features for gaining information has been applied not only with the HTTP protocol but also with other protocols. For example, Song et al. [8] showed that some SSH variants are not secure. They showed that simple statistical analysis was able to reveal sensitive information such as login passwords. Additionally, they showed that advanced statistical analysis on timing information can reveal what users type. Another example of a protocol that was shown to be vulnerable is Voice Over IP (VoIP). Wright et al. [14] showed that it is possible to identify spoken phrases by using encrypted VoIP packet length, when variable bit rate (VBR) encoding is used. They used a Hidden Markov model that achieved more than 90% recall and precision. Conti et al. [5] presented various action classifications for a range of applications for mobile devices that achieve high accuracy.

This paper’s main contributions are:

- This is the first work that shows how to identify the user’s operating system, browser and application from his HTTPS traffic. Inspired by other works presented above, we exploit traffic patterns. Additionally, we present new features that exploit browsers’ bursty behavior and SSL behavior. Using the baseline features, the accuracy is 93.51%, while using a combination of baseline and new features achieves accuracy of 96.06%.
- We provide a comprehensive dataset that contains more than 20000 labeled sessions. The operating systems in the dataset are: Windows, Linux-Ubuntu and OSX. The browsers are: Chrome, Internet Explorer, Firefox and Safari. The applications are: YouTube, Facebook and Twitter. The dataset is available for download at [52].

II. IDENTIFICATION OF USER’S OPERATING SYSTEM, BROWSER AND APPLICATION

The goal of this paper is to identify user operating system, browser and application. To achieve this goal, we use supervised machine learning techniques. Supervised machine learning techniques learn a function that given a sample returns a label. The learning is carried out using a dataset of labeled samples. In our case, we chose to use sessions as
samples where a session is the tuple <Protocol, IP source, IP destination, Port source, Port destination> and the label is the tuple <OS, Browser, Application>. Thus, our task is inherently a multiclass learning with 30 classes (see Figure 1a for the labels and their statistics in the dataset).

The rest of this section is organized as follows: In section II-A we describe how we collected the dataset and the dataset characteristics. In section II-B we describe and discuss our feature extraction scheme. Finally, in section II-C we describe the machine learning methodology we used.

A. Dataset

We used the Selenium web automation tool [53] to develop crawlers for gathering the dataset. We gathered all the traffic that passed through port 443 (TLS/SSL). Finally, we split the traffic into sessions using SplitCap [54].

For YouTube and Facebook traffic, we used the crawler on a standard internet connection over various operating systems and various browsers and combinations thereof. For Facebook, the same account was used both for sending and receiving posts. For Twitter, we had one sending account and several receiving accounts (followers) where they ranged over various operating systems and various browsers and combinations thereof. Teamviewer’s traffic was generated by us actively without a crawler. In addition to our active traffic, we also observed background traffic that operating systems, browsers and applications created (Google-Background, Microsoft-Background). Dropbox traffic was composed both of active (no crawler) and background traffic.

Any traffic that we could not recognize was labeled as unknown. The browser label part of the tuple of stand alone applications which do not work under a browser (e.g. Dropbox, Teamviewer) were labeled as Non-Browser.

The dataset was collected over the period of more than two months in our research lab over diverse connections (wired and WiFi) and networks conditions (over workdays and weekends 24/7).

Our dataset contains more than 20000 sessions. The tuple labels statistics can be seen in Figure 1a. Operating system, browser, application statistics can be seen in Figures 1b,1c,1d correspondingly.

B. Feature Extraction

This section describes how we extract features from a session and the feature characteristics. Encrypted traffic generally relies on SSL/TLS for secure communication. These protocols are built on top of the TCP/IP suite. The TCP layer receives encrypted data from the above layer and divides data into chunks if the packets exceed the Maximum Segment Size (MSS). Then, for each chunk it adds a TCP header creating a TCP segment. Each TCP segment is encapsulated into an Internet Protocol (IP) datagram. As TCP packets do not include a session identifier, we identify a session using the tuple <Protocol, IP source, IP destination, Port source, Port destination>.

A session contains two flows: forward and backward. A flow is defined as time ordered sequence of TCP packets during a single TCP session. The forward flow is defined as a time series bytes transported by incoming packets only, while the backward flow is defined as a time series bytes transported by
outgoing packets only. We use the forward, backward and their combination as a representation of a connection. Additionally we also use time series features such as inter arrival time differentials between different packets on the same flow.

The feature extraction takes as an input the session network traffic and extracts features from it. In this paper, we consider two sets of features and their combination. First, a typical feature set, used in many traffic classification methods [33]–[38], [50], which we call “base features” is presented in Table Ia.

Second, we present a new set of features in Table Ib, which we call “new features”. This set of features is based on a comprehensive network traffic analysis, in which we tried to identify traffic parameters that differentiate between different operating systems and browsers. Our new set is robust to changes (not based on a small set of strong parameters or fingerprint). The set of features include new SSL features, new TCP features and the bursty behavior of the browsers (peaks) which is defined as a section of traffic where there is silence before and after. This section of traffic is called a peak. An example of the bursty behavior of browsers is depicted in Figure 2. Our previous works also used these peak features to classify video titles [55] and quality representation [56]. Note that, the bursty behavior of browser traffic was observed for YouTube traffic in [57], [58].

C. Learning

In this section we describe our machine learning methodology. We chose to use the Support Vector Machine (SVM) [59] with Radial Basis Function (RBF) as the kernel function because it has excellent performance in many machine learning applications. We trained and tested on 70% train and 30% test splits five times and accuracy is reported as the average of these experiments.

First, features were scaled between zero and one at training and the same scaling factors were used for the test set. 5-fold cross validation was used for choosing both the regularization parameter of SVM, C, over the set \( \{2^{-5}, 2^{-3}, \ldots, 2^{15}\} \) and for the gamma parameter of RBF, over the set \( \{2^{-15}, 2^{-13}, \ldots, 2^3\} \). We used LIBSVM [60] to train and test our data set.
Fig. 3: Accuracy results for SVM-RBF with different features set. This work is the first to show that the tuple <OS, Browser, Application> classification is possible with high accuracy. Note that, naive classification based on the data set statistics will have only 32.34% accuracy (<Windows, IExplorer, Twitter>). Moreover, adding our new features increase the accuracy to 96.06%.

Fig. 4: Confusion matrices (rows are ground truth). For most tuples the classification is almost perfect. Exceptions happens mostly between similar tuples and the unknown classes (which can actually be a correct answer that we cannot verify). For example, “Ubuntu Chrome Google-Background” is mistakenly classified as “Ubuntu Chrome Unknown” in 18% of the cases and “Ubuntu Firefox Google-Background” in 7%. The total accuracy is to 96.06%.
[57] Pablo Ameigeiras, Juan J. Ramos-Muoz, Jorge Navarro-Ortiz, and
Juan M. Lpez-Soler. Analysis and modelling of youtube traffic. TETT,
2012.
[58] Ashwin Rao, Arnaud Legout, Yeon-sup Lim, Don Towsley, Chadi
Barakat, and Walid Dabbous. Network characteristics of video streaming
traffic. In CoNEXT, 2011.
[59] C. Cortes and V. Vapnik. Support vector machine. Machine learning,
1995.
[60] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support
vector machines. IST, 2011.