

Robust Machine Learning for Encrypted Traffic Classification

Amit Dvir^{*†}, Yehonatan Zion^{*‡}, Jonathan Muehlstein^{*‡}, Ofir Pele^{†‡} Ran Dubin[‡],

^{*} Department of Computer Science, Ariel University, Israel

[†] Department of Electrical and Electronics Engineering, Ariel University, Israel

[‡] Ariel Cyber Innovation Center, Ariel University, Israel

Abstract—Desktops and laptops can be maliciously exploited to violate privacy. In this paper, we consider the daily battle between the passive attacker who is targeting a specific user against a user that may be adversarial opponent. In this scenario, while the attacker tries to choose the best vector attack by surreptitiously monitoring the victim’s encrypted network traffic in order to identify user’s parameters such as the Operating System (OS), browser and apps. The user may use tools such as a Virtual Private Network (VPN) or even change protocols parameters to protect his/her privacy. We provide a large dataset of more than 20,000 examples for this task. We run a comprehensive set of experiments, that achieves high (above 85%) classification accuracy, robustness and resilience to changes of features as a function of different network conditions at test time. We also show the effect of a small training set on the accuracy.

Index Terms—Passive Attacker, Traffic analysis, HTTPS, Adversarial Opponent

I. INTRODUCTION

Today, the amount of sensitive data that users handle with devices is growing rapidly. Typical users access their bank web-site, social network web-sites and get information about diseases, sexual or religious preferences that are assumed to be confidential and non-hackable.

Personal devices can be attacked in two broad ways which are known as passive and active. Active adversaries try to physically or remotely control the user’s device. Passive adversaries may violate the privacy of the user by sniffing the network traffic of the devices from the network side. A passive adversary can use this information to map user actions and obtain information on personal interests and habits. If the network traffic is not encrypted, a passive attacker can simply analyze the unencrypted HTTP requests and responses [1]–[4]. However, most Internet traffic today is encrypted as a result of increased user awareness of privacy threats and encouragement by Google to get all website owners to switch from HTTP to HTTPS by indicating whether sites use secure, encrypted connections in Google ranking algorithms [5], [6]. One immediate outcome has been that traditional Deep Packet Inspection (DPI) methods for information retrieval are no longer viable.

Nevertheless, numerous studies have shown that encryption is not sufficient to protect confidentiality [7]–[46]. Bujlow et al. [30] presented a survey on popular DPI tools for traffic classification. Nguyen and Armitage [31] surveyed machine learning techniques for Internet traffic classification.

Moore et al. [36] used a Naïve Bayes classifier, a supervised machine learning approach to classifying Internet traffic applications. Williams et al. [38] conducted a comparison of five machine learning algorithms that were used to classify Internet traffic applications. Auld et al. [37] proposed using a supervised Bayesian neural network to classify Internet traffic applications. Alshamarri et al. [39] compared AdaBoost, Support Vector Machines, Naïve Bayes, RIPPER and C4.5 classification accuracy of Skype traffic. Donato et al. [42] presented a method for application classification called the Traffic Identification Engine. Niemczyk et al. [41] suggested dividing the session into time buckets (10 seconds). They conclude that the new feature (buckets) was almost perfect for Skype but not able to differentiate between browsers and between joint application and browser usage.

In this work, we consider the daily battle between the passive attacker who is targeting a specific user which in some cases the user may be adversarial opponents (e.g., [47]). In this scenario, while the attacker tries to choose the best vector attack against the user by surreptitiously monitoring the victim’s network traffic to identify users parameters such as the Operating System (OS), browser and apps, the user may use tools such as a Virtual Private Network (VPN) or even change protocols parameters to protect his/her privacy. The attacker can either sniff the wireless network of the user or can sniff the user traffic over the ISP network (e.g., government). Based on the vulnerabilities of the tuple (OS, Browser, Applications), the attacker will choose the best malwares or Advanced Persistent Threat (APT) method and due to the encryption the attacker may consider using machine learning to classify those user parameters (OS, browser and applications).

Although studies such as the above have shown that encryption is not sufficient and presented systems to identify and classify user parameters such as the operating system (e.g. [48]–[50]), the user browser (e.g. [50]) and the user applications (e.g. [35], [39]). None of them have dealt with the case where the user may be aware of the classification system and decides to become an adversarial opponent. In this case, the user will try to manipulate the data to protect privacy by changing the network traffic or the protocols parameters. Due to the fact that the passive attacker may use machine learning algorithms, the user may change his traffic during testing time and not in training which increases the difficulty of the passive attacker to classify the user parameters.

To respond to this need we first present a machine learning

algorithms, for the passive attacker, that uses new features to classify the user tuple (OS, Browser, Application). Second, we show the performance of our machine learning (robustness) to attacks by adversarial opponent that implement tools such as VPN which extends a private network. Third, we show that our system is also robust against changes in protocol parameters (e.g. cipher suite). The last two achievements are the main contribution of the paper. Finally, we show that our algorithm achieves high accuracy even when using a small number of training samples or taking only a short session time as a test sample.

In summary:

- We present a machine learning system to identify the user’s operating system, browser and application from HTTPS traffic which achieves 96.06% accuracy. Our machine learning use new features that exploit browsers’ bursty behavior and SSL behavior.
- Our system only incurs marginal damage in the case of adversarial opponent. Where the adversarial opponent may change the protocol parameters, use network tools or shortage the information (small amount of session packets). We show that our solution is robust to changes in cipher suites (protocol changes), 94%, and VPN (network tool), 83%, where both changes are *only* on the testing data (i.e. the user changed the parameters whereas the machine learning training and classification did not change). We also show that using small training datasets only incurs marginal damage. For example, using only 500 examples for training (vs. the full training dataset of $0.7 \times 20,633$) we achieved a reasonable accuracy of 85%.
- We investigate the resilience of the system to short session times; for example, using only the first 10 seconds of a session for training and testing. We show that although we use shorter sessions which yields less meta data, we achieve 94.2% accuracy. Note that, short session times show that our system can be a real time system (do not need to wait to the end of the session).
- We provide a comprehensive dataset that contains more than 20,000 labeled sessions. The operating systems tested were: Windows, Linux-Ubuntu and OSX. The browsers tested were: Chrome, Internet Explorer, Firefox and Safari. The applications tested were: YouTube, Facebook, Twitter, Teamviewer, Skype and Dropbox. The dataset is available for download at [51] and the codes for the crawler, feature extractions and machine learning algorithms can be found in [52]–[54].

In this paper, we extend our previous work [55] on machine learning algorithms that classify tuples. In this work, we investigated the robustness and resilience of our system against adversarial opponents users. These users attempting to protect their privacy and are aware of the fact that their privacy may been infringed. We also introduce two machine learning algorithms and extended datasets and discuss possible countermeasures of our system.

The remainder of this paper is organized as follows. In Section II we review the state of the art on this topic. Section III presents our solution to identifying the user’s operating

system, browser and application. In Section IV we evaluate our method including its robustness to changes in network conditions at test time. In Section V we discuss limitations and possible countermeasures. Finally, we discuss future lines of work in Section VI.

II. RELATED WORKS

Feature extraction methods for traffic classification, application classification and application analysis include session duration [39], number of packets in a session [35], [56], minimum, maximum and average values of inter-arrival packets time [35], [39], payload size information [35], bit rate [57], [58], round-trip time [57], packet direction [59], SSL parameters [46] and server sent bit-rate [60].

Liberatore and Levine [61] showed the effectiveness of two traffic analysis techniques for the identification of encrypted HTTP streams. One is based on a naïve Bayes classifier and the other on Jaccard’s coefficient similarity measure. They also proposed several methods for actively countering these techniques. They found these methods to be effective, albeit at the cost of a significant increase in the size of the traffic stream. Panchenko et al. [62] 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 [63]. Cai et al. [64] presented a web page fingerprinting attack and showed that it is able to overcome defenses such as the application-level defense HTTPoS [65] and randomized pipelining over Tor.

Mobile devices, which have different operating systems and different applications implementation leading in many cases to different network behavior, are also susceptible to attackers seeking information on user privacy, applications [66]–[72] and actions [8], [73], [74]. Saltaformaggio et al. [73] presented NetScope, a passive framework for identifying user activities within the wireless network traffic based on inspecting IP headers. Conti et al. [7], [8], [74], [75] devised a highly accuracy classification frameworks for various user mobile actions and applications using network features such as size, direction (incoming/outgoing), and timing.

Gathering information on the Operating System (OS) of the user can be useful too. Passive sniffing of the OS fingerprinting techniques was proposed in [48], [49], [76], [77]. p0f [48] is a well-known and widely used tool that uses header fields such as the TCP SYN and SYN-ACK exchanges to fingerprint the OS. However, any change in the 3-way handshake of the TCP will affect the tool. Anderson and McGrew [49] presented an effective approach to OS passive fingerprinting that uses a combination of encrypted data (TLS+TCP/IP) with non-encrypted data (HTTP) over multi-sessions. Although certain papers [48], [49] classified the operating system, they based their classification on either protocol parameters (TCP handshake) [48] or on non-encrypted data [49]. Moreover, none of them considered the case of adversarial opponents that try to protect against the classification.

Passive fingerprinting of browser clients from encrypted traffic is also necessary. Husak et al. [50] proposed real-time exact pattern matching for the identification of the user’s OS

or browser based on SSL/TLS fingerprinting. In this work [50], the system has to identify the SSL parameters and is not robust to changes in the SSL parameters such as cipher suite parameters by adversarial opponents (the user).

Machine learning systems offer unparalleled flexibility in dealing with evolving input in a variety of applications. However, machine learning algorithms themselves can be a target of attack by a malicious adversary. Barreno et al. [78] described different types of attacks on machine learning and a variety of defenses against these attacks. Demontis et al. [79] presented a simple and scalable secure-learning paradigm that mitigates the impact of invasion attacks, while only slightly affecting the detection rate. In our paper attacks on machine learning can be by using network tools such as VPN or changing the protocol parameters.

III. ROBUST IDENTIFICATION OF USERS' OPERATING SYSTEM, BROWSER AND APPLICATION

The goal of this paper is to identify the user operating system, browser and application even in the case where the user is aware of the system and may try to decrease the accuracy of the system by changing protocol parameters and/or by using network tools. To achieve this goal, we used supervised machine learning techniques.

Supervised machine learning techniques learn a function which given a sample returns a label. 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 form of multiclass learning with more than 30 classes (see Figure 2a for the labels and their statistics in the dataset).

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

A. Dataset

In order to simulate different usage scenarios, we built an automatic system that emulates user browsing (websites and associated actions). We used the Selenium web automation tool [80] to develop crawlers to gather the dataset. We collected all the traffic that passed through port 443 (TLS/SSL). Finally, we split the traffic into sessions using SplitCap [81].

We used the crawlers on a standard Internet connection over various operating systems and various browsers and combinations thereof. The passive users follow the active user and all hosts capture traffic. The dataset was collected automatically 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). Figure 1 depicts the lab architecture (8 computers, with various operation systems and browsers) where each entity ran a crawler; the crawler code can be found in [52].

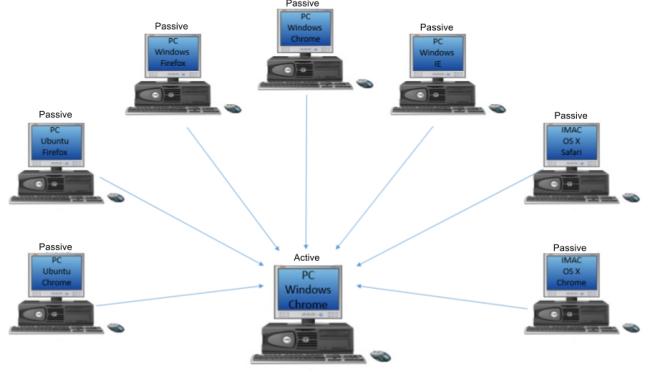


Fig. 1: The Architecture of the Laboratory

The crawler covered multiple sequences of UI actions across different systems and configurations. This presented various challenges. First, we were faced with an unclear workflow when we had to add features to the code base. To handle this issue we defined a configuration file with all the required simulation information. The configuration file was stored in a JSON file format. This method allowed us to abstract the core functionality code from the configuration of each computer and made it easier to handle bugs and add features.

Second, the crawler had to work on several different applications and functions both as an active user and as well as a passive user. To do so, we made the crawler generic such that for each application all we needed to do was add the XPATHs (XPATH is a query language for node selection from within an XML document.) of the fields of each action and run it. This idea was good in theory but involved technical complexity that wasted too much time. Thus, we developed a crawler for each application.

Third, the crawler had to handle application updates both in the context of security as well as the UI format. It makes sense that services like YouTube, Facebook and Twitter will change the UI of their services. It is also reasonable that from time to time they will change their privacy and security policy. Changes such as these gave us considerable trouble since each change in the UI affected the HTML design and hence hindered our crawler's effectiveness since the crawler relies on the HTML structure.

Finally, we had to deal with the way web services handle bots like our own. To prevent abusive behavior, social media platforms enforce timing constraints such as X posts per day and a single new post per Y seconds. To overcome this issue we experimented with multiple timing configurations.

For Facebook, the same account was used for both sending and receiving posts. For Twitter, we had one sending account and several receiving accounts (followers) which ranged over various operating systems and various browsers and combinations thereof. Teamviewer traffic was generated by us actively without a crawler. In addition to our active traffic, we also observed background traffic that the operating systems, browsers and applications created (Google-Services, Microsoft-Services). One example of a service is Google Analytics or Microsoft Live. Dropbox traffic was composed both

of active (no crawler) and background traffic. Any traffic that we could not identify was labeled unidentified. 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.

Thus overall our dataset contained more than 20,000 sessions. The average duration of a session was 518 seconds where on average each session had 520 forward packets (the average forward traffic size was 261 Kbytes) and 637 backward packets (average backward traffic size is 615 Kbytes). The tuple label statistics can be seen in Figure 2a. The operating system, browser and application statistics can be seen in Figures 2b,2c,2d respectively. The dataset can be found in [51].

B. Feature Extraction

Using raw data in a machine learning method is problematic because the data are non-structured and contain redundant information. Thus, there is a need to build a structured representation of the raw data that is informative as to the specific problem domain. Building this representation is called feature extraction [82, Chapter 5.3].

We extracted features from a session of encrypted traffic which 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 the 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 identified 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 a time ordered sequence of TCP packets during a single TCP session. The forward flow is defined as a time series of bytes transported by incoming packets alone, whereas the backward flow is defined as a time series of bytes transported solely by outgoing packets. We used the forward and backward flows and their combination as a representation of a connection. Additionally, we also used time series features such as the inter-arrival time differentials between different packets on the same flow. Based on our previous work [83], for classifying video titles, we also used the bursty behavior of the browsers (peaks) which is defined as a section of traffic preceded and followed by silence. An example of the bursty behavior of browsers is depicted in Figure 3. Note that the bursty behavior of browser traffic was also observed for YouTube traffic in [84], [85]. The feature extraction takes the session network traffic as input and extracts features from it.

This paper proposes a new set of features incurred by browser bursty behavior and SSL behavior. We combine our new feature set with the previous works features (summary of features of the papers presented in Section II). In the results section we show that this combination (named as Combined set) considerably outperforms previous works (base-line set, termed the common set). We consider nine sets of features as

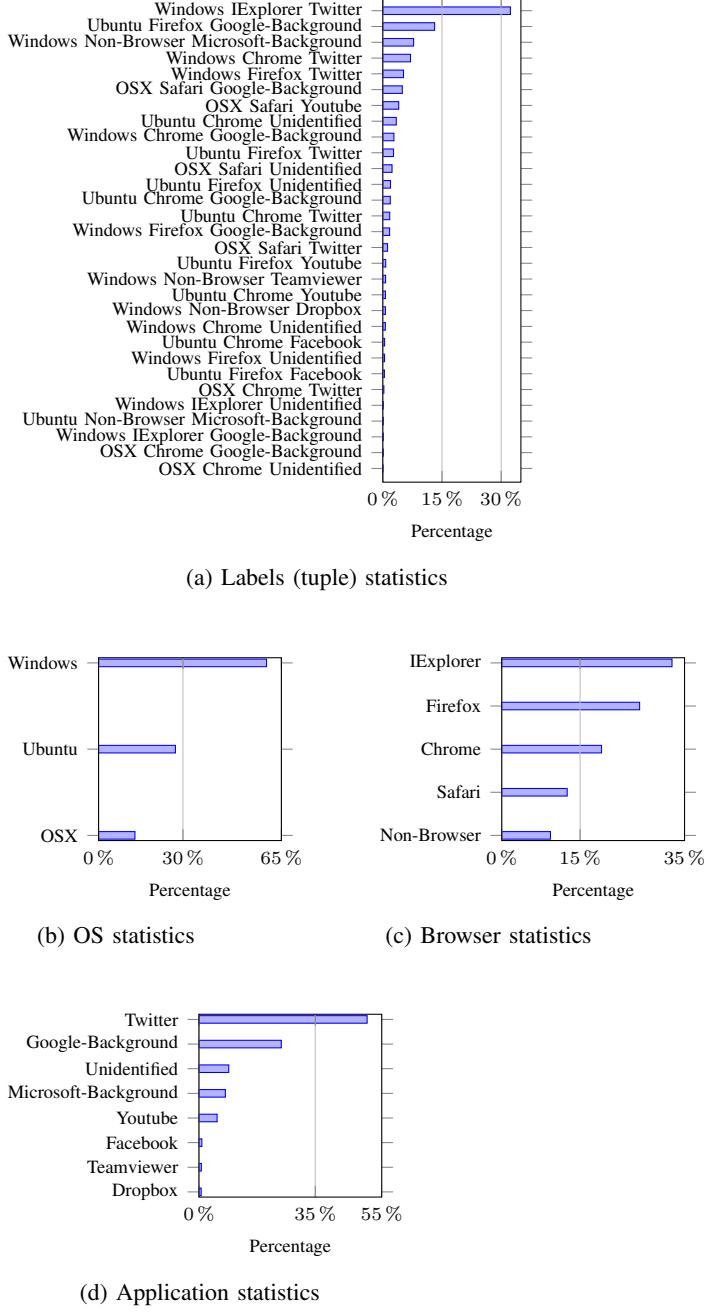


Fig. 2: Dataset statistics

can be seen in Table II. The feature extraction code can be found in [53].

C. Learning

In this section we describe our machine learning methodology. Supervised classification learning methods learn a classification function from a set of pre-labeled examples. The classification function is then used for classifying unseen test examples. There are two types of supervised classification learning methods: lazy and eager. Lazy learning algorithms store the training data as is and then apply a classification function on a new test example where the classification function is parameterized with the pre-labeled training examples. Eager

Forward packets
Forward total bytes
Min forward inter arrival time difference
Max forward inter arrival time difference
Mean forward inter arrival time difference
STD forward inter arrival time difference
Mean forward packets
STD forward packets
Backward packets
Backward total bytes
Min backward inter arrival time difference
Max backward inter arrival time difference
Mean backward inter arrival time difference
STD backward inter arrival time difference
Mean backward packets
STD backward packets
Mean forward TTL value
Minimum forward packet
Minimum backward packet
Maximum forward packet
Maximum backward packet
Total packets
Minimum packet size
Maximum packet size
Mean packet size
Packet size variance

(a) common features

TCP initial window size
TCP window scaling factor
SSL compression methods
SSL extension count
SSL cipher methods
SSL session ID len
Forward peak MAX throughput
Mean throughput of backward peaks
Max throughput of backward peaks
Backward min peak throughput
Backward STD peak throughput
Forward number of bursts
Backward number of bursts
Forward min peak throughput
Mean throughput of forward peaks
Forward STD peak throughput
Mean backward peak inter arrival time diff
Minimum backward peak inter arrival time diff
Maximum backward peak inter arrival time diff
STD backward peak inter arrival time diff
Mean forward peak inter arrival time diff
Minimum forward peak inter arrival time diff
Maximum forward peak inter arrival time diff
STD forward peak inter arrival time diff
Keep alive packets
TCP Maximum Segment Size
Forward SSL Version

(b) new features

TABLE I: The common features used in previous traffic classification methods and the new features proposed in this paper.

learning algorithms, on the other hand, carry out a learning process on the pre-labeled training examples. Eager learning algorithms often perform better since the offline learning stage increases robustness to noise.

The lazy machine learning algorithm we selected is the nearest neighbor algorithm [86]. In this algorithm, the classification function computes similarities between a new test sample and

Common Feature Set	a set of features used in previous traffic classification methods [36]–[41], [77] (presented in Table Ia). This feature set includes the common features in previous papers which we used as our baseline, and as a comparison to our features set.
Peaks Feature Set	Only the bursty behavior of the browsers. Can be useful in the case a smart user (adversarial opponent) changes all the protocols parameters.
New Feature Set	A new set of features (presented in Table Ib), based on a comprehensive network traffic analysis, in which we identified traffic parameters that differentiate between different operating systems and browsers. The set of features included new SSL features, new TCP features and the bursty behavior of the browsers (peaks) which is defined as a section of traffic preceded and followed by silence.
Common Stats Feature Set	Only statistics parameters from Table Ia
Statistics	Only statistics parameters from Tables Ia, Ib. Can be useful in the case a smart user changes all the protocols parameters and we can only use the parameters related to the arriving packets.
Combined Feature Set	All available features combined (all the sets above).
Combined no peaks Feature Set	All available features combined minus the peak features. Can be useful in the case a smart user changes the traffic pattern of the session.
Combined no SSL Feature Set	All available features combined minus the SSL related features. Can be useful in the case a smart user changes the SSL parameters (in the header, e.g. Cipher Suite).
Combined no TCP Feature Set	All available features combined minus the TCP protocol related features. Can be useful in the case a smart user changes the TCP parameters (in the header, e.g. MSS size).

TABLE II: Feature Sets

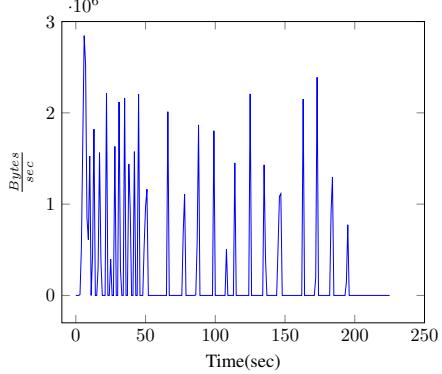


Fig. 3: An example of bursty behavior of browser traffic.

all pre-labeled examples. The test sample is then assigned to the class of the most similar example from the training data.

The first eager machine learning algorithm we chose was the Support Vector Machine (SVM) [87]. The binary linear SVM models training examples as points in space, and then divides the space using a hyperplane to give the best separation between the two classes. We used the LIBSVM package [88] which implements the one-vs-one multiclass scheme. We used three versions of the SVM:

- SVM+RBF - SVM with the Radial Basis Function (RBF) as the kernel function.
- SVM+SIM - SVM with threshold similarities to the training samples as features [89]. Similarity is bounded

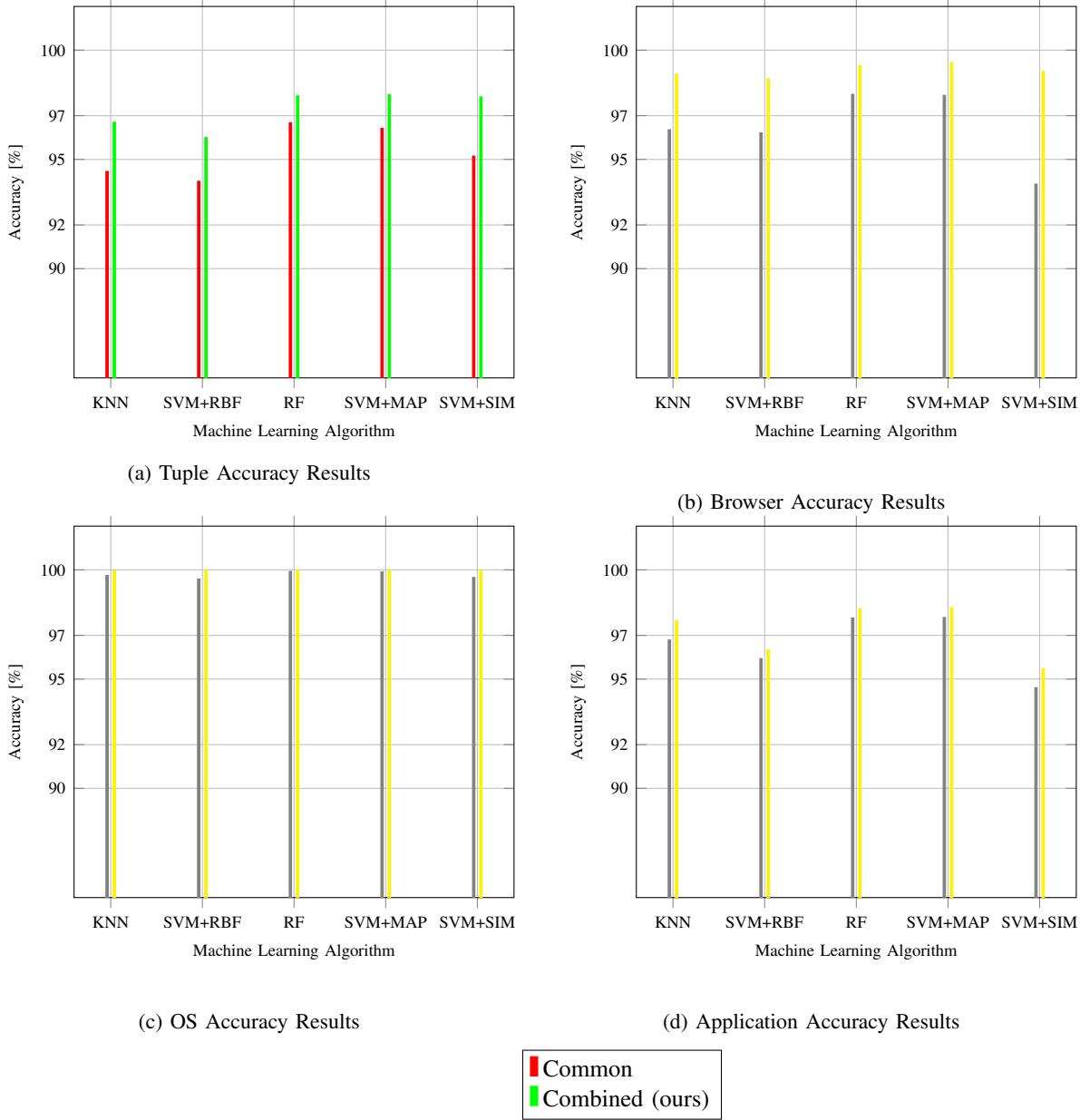


Fig. 4: Accuracy results for KNN, SVM-RBF, SVM-MAP, SVM-SIM, RF. Adding our new features increased the accuracy to 96.6%.

by a threshold. This is a heuristic based on the assumption that dissimilar samples add noise rather than information. The threshold similarity value and the distance function are chosen by the cross-validation process.

$$u, v \in \text{samples}, 1 - \frac{\min(\text{distance}(u, v), \text{threshold})}{\text{threshold}}$$

- SVM+MAP - SVM with modified RBF similarities as features [89] where the similarity was the same as in the RBF function except for the distance function which was not necessarily squared Euclidean. The distance function and gamma values are chosen by cross-validation process.

$$u, v \in \text{samples}, \exp(-\gamma \cdot \text{distance}(u, v))$$

The second eager machine learning algorithm we chose was the Random Forest algorithm [90], [91]. The Random Forest algorithm grows multiple trees by randomly selecting subsets of features. That is, trees are constructed in random subspaces [90], [91].

Thus in total we had five machine learning algorithms. We fine-tuned the hyper-parameters of our machine learning algorithms through a grid search procedure combined with 5-fold cross validation over the training set. For all the machine learning algorithms, features were scaled between zero and one at training and the same scaling factors were used for the test set. A one-tailed t-test with a p-value 0.05 for every pair of algorithm results vector was conducted to test whether the result of one algorithm was significantly better than the others.

Real labels

	Predicted labels																											
Real labels	Windows IEExplorer Twitter	Ubuntu Firefox Google-Services	Windows Non-Browser Microsoft-Services	Windows Chrome Twitter	OSX Safari Google-Services	Ubuntu Chrome Unidentified	Windows Chrome Google-Services	Ubuntu Firefox Twitter	OSX Safari Unidentified	Ubuntu Firefox Unidentified	Ubuntu Chrome Google-Services	Ubuntu Firefox Twitter	Ubuntu Chrome Twitter	Windows Firefox Google-Services	OSX Safari Twitter	Ubuntu Firefox YouTube	Windows Non-Browser Teamviewer	Ubuntu Chrome YouTube	Windows Non-Browser Dropbox	Windows Chrome unidentified	Ubuntu Chrome Facebook	Windows Firefox unidentified	Ubuntu Firefox Facebook	OSX Chrome Twitter	Ubuntu Non-Browser Skype	Windows IEExplorer Google-Services	OSX Chrome Google-Services	OSX Chrome Unidentified
Windows IEExplorer Twitter	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Ubuntu Firefox Google-Services	0	.97	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Windows Non-Browser Microsoft-Services	0	.99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Windows Chrome Twitter	0	0	.99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Windows Firefox Twitter	0	0	0	.98	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
OSX Safari Google-Services	0	0	0	0	.92	.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
OSX Safari YouTube	0	0	0	0	0	.97	.01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Ubuntu Chrome Unidentified	0	0	0	0	0	0	.84	0	0	0	0	.07	.04	0	0	0	0	.01	0	.03	0	0	0	0	0	0		
Windows Chrome Google-Services	0	0	.01	.03	0	0	0	.94	0	0	0	0	0	.02	0	0	0	0	0	.01	0	0	0	0	0	0		
Ubuntu Firefox Twitter	0	0	0	0	0	0	0	.95	0	.03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
OSX Safari Unidentified	0	0	0	0	0	.06	.01	0	0	.91	0	0	0	0	.01	0	0	0	0	0	0	0	0	0	0	0		
Ubuntu Firefox Unidentified	0	.02	0	0	0	0	0	.08	0	.87	0	0	0	.73	0	0	0	0	.02	0	0	.03	0	0	0	0		
Ubuntu Chrome Google-Services	0	.07	0	0	0	0	.18	0	0	0	.03	.84	0	0	0	0	0	.01	0	0	.01	0	0	0	0	0		
Ubuntu Chrome Twitter	0	.02	0	0	0	0	.08	0	0	0	.03	.84	0	0	0	0	.01	0	0	.01	0	0	0	0	0	0		
Windows Firefox Google-Services	0	0	0	.01	0	0	0	.01	0	0	0	0	.97	0	0	0	0	0	0	.01	0	0	0	0	0	0		
OSX Safari Twitter	0	0	0	0	0	.06	0	0	.03	0	0	0	.91	0	0	0	0	0	0	0	0	0	0	0	0	0		
Ubuntu Firefox YouTube	0	.02	0	0	0	0	0	.02	0	.02	0	0	.93	0	0	0	0	0	0	0	0	0	0	0	0	0		
Windows Non-Browser Teamviewer	0	0	0	0	0	0	0	0	0	0	0	0	.1	0	0	0	0	0	0	0	0	0	0	0	0	0		
Ubuntu Chrome YouTube	0	0	0	0	0	0	.07	0	0	0	.13	.04	0	0	0	.74	0	0	.02	0	0	0	0	0	0	0		
Windows Non-Browser Dropbox	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.1	0	0	0	0	0	0	0	0	0		
Windows Chrome Unidentified	0	0	.02	.09	0	0	0	.02	0	0	0	0	0	0	0	0	.86	0	0	0	0	0	0	0	0	0		
Ubuntu Chrome Facebook	0	0	0	0	0	0	.3	0	0	0	.04	0	0	0	0	0	.67	0	0	0	0	0	0	0	0	0		
Windows Firefox Unidentified	0	0	.06	0	0	0	0	0	0	0	0	0	0	0	0	0	.94	0	0	0	0	0	0	0	0	0		
Ubuntu Firefox Facebook	0	.06	0	0	0	0	0	0	.11	.28	0	0	0	0	0	0	.56	0	0	0	0	0	0	0	0	0		
OSX Chrome Twitter	0	0	0	0	0	0	.13	0	0	0	0	0	0	0	0	0	.75	0	0	0	.06	.06	0	0	0	0		
Windows IEExplorer Unidentified	.71	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.29	0	0	0	0	0	0	0	0		
Ubuntu Non-Browser Skype	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Windows IEExplorer Google-Services	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
OSX Chrome Google-Services	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
OSX Chrome Unidentified	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Fig. 5: Confusion matrix (rows are ground truth). For most tuples the classification is almost perfect. Exceptions occur mostly between similar tuples and the Unidentified classes (which may actually be a correct answer that we could not verify). For example, “Ubuntu Chrome Google-Services” was mistakenly classified as “Ubuntu Chrome Unidentified” in 18% of the cases and “Ubuntu Firefox Google-Services” in 7%.

For the KNN algorithm, we used cross validation to choose the number of neighbors over the set $\{4, 6, \dots, 20\}$, the uniform or distance-based weights, and the distance measures: Euclidean, Manhattan, Chebyshev, Hamming and Canberra. For the SVM algorithm, we used cross validation to choose both the regularization parameter of SVM, C , over the set $\{2^{-5}, 2^{-3}, \dots, 2^{15}\}$ and for the gamma parameter of RBF, over the set $\{2^{-15}, 2^{-13}, \dots, 2^3\}$. We used LIBSVM [88] to train and test our dataset. For the Random Forest algorithm we used cross validation to choose the number of trees over the set $\{20, 40, \dots, 120\}$. The machine learning algorithm code can be found in [54].

IV. RESULTS

We trained and tested on 70% train and 30% test splits five times; accuracy is reported as the average of these experiments (average error bar is 0.01%). We first show that the accuracy of the tuple classification using our combined set is higher compared to the baseline feature set (common set). Then we present our main innovation; that even a smart user that is aware of our machine learning system and tries to manipulate the system by changing the protocols parameters or using other tools has a negligible effect on the accuracy of our system. Then, we demonstrate the robustness of the system against a smart user, in the case where we cannot sniff for a lengthy period of time the input of the training set or when taking training samples limited by time (the first x seconds/minutes of the sessions). To assess whether our system can run on real

time system, we show that the training and testing time is low (test time in milliseconds).

The accuracy for the tuple $\langle OS, \text{Browser}, \text{Application} \rangle$ classification with our feature set as compared to the baseline feature set (common set) is presented in Figure 4. The figure shows that the tuple $\langle OS, \text{Browser}, \text{Application} \rangle$ classification of encrypted classification is possible with high accuracy. In all the experiments, using our combined set achieved the best results whereas the Random Forest algorithm and SVM+MAP both achieved the highest accuracy. For tuple classification, the addition of our new features increased the accuracy from 93.52% to 96.06%. A confusion matrix for the tuple accuracy is shown in Fig 5. It shows that the classification is almost perfect, and most of the mistakes could be ascribed to unidentified labels which may have been be a correct answer that we could not verify.

The accuracy for the tuple $\langle OS, \text{Browser}, \text{Application} \rangle$ classification with the nine feature sets is presented in Figure 6. Note that some of the feature sets (e.g. combined without TCP) are equivalent to the case where adversarial opponents change the protocols parameters. Using our base + new features achieved the best results whereas the Random Forest algorithm and SVM+MAP both achieved the highest accuracy. When using a subset of the features to mitigate the influence of adversarial opponent also achieved high accuracy (between 94% – 95.8%).

The influence of using network tools by the user (e.g. VPN) to affect our machine learning system can be seen in Figure

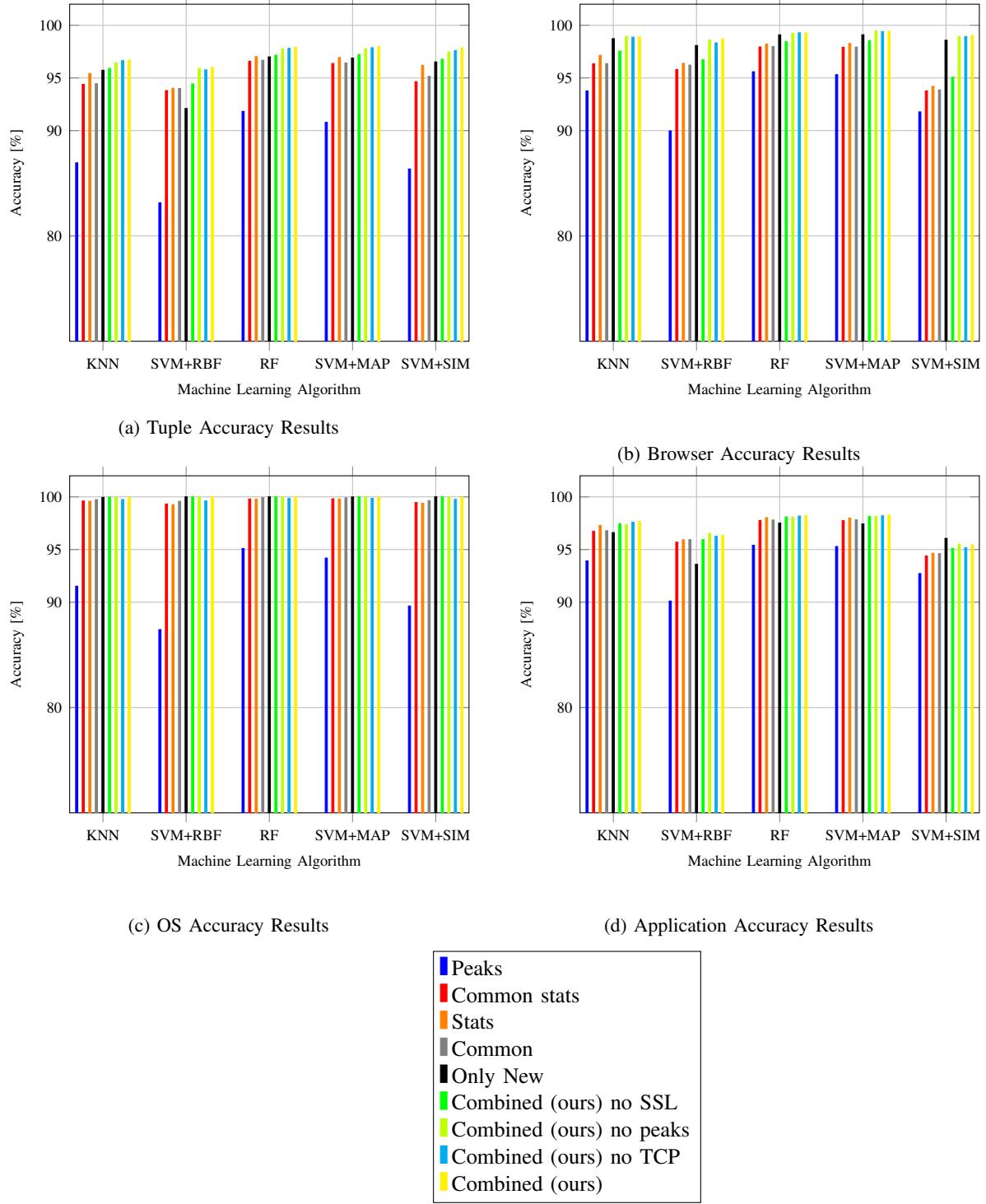


Fig. 6: Accuracy results for KNN, SVM-RBF, SVM-MAP, SVM-SIM, RF with different features sets that are equivalent to adversarial opponents.

7a. Although the opponent aggregates all the sessions together, our system is still able to classify the operation system and the browser with good accuracy (81%). When classifying both the OS and the browser, the best performance is achieved using the SVM+MAP algorithm with combined feature set. Most other combinations exhibited relatively low performance.

The impact on accuracy when the user changes protocol

parameters such as Cipher suites (SSL protocol parameter) can be seen in Fig. 7b. Although the opponent changes the number and the value of the cipher suites in the SSL header, our system is still able to classify the operation system and the browser with satisfactory accuracy (91%). When classifying both the OS and the Browser, the best performance was achieved using the RF algorithm with combined features set. Note that we

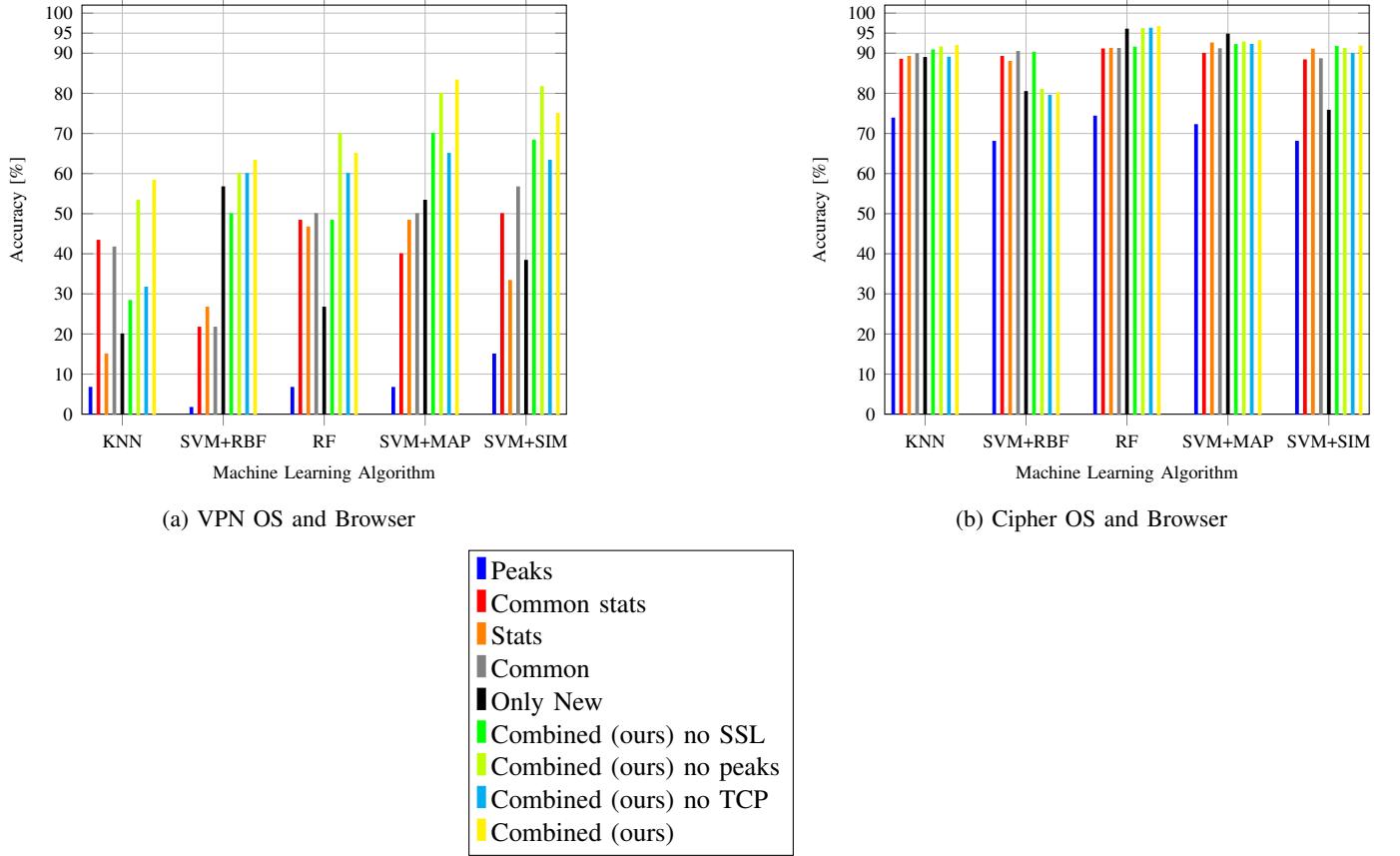


Fig. 7: The influence of using VPN or changing the Cipher suite at test time on the accuracy results

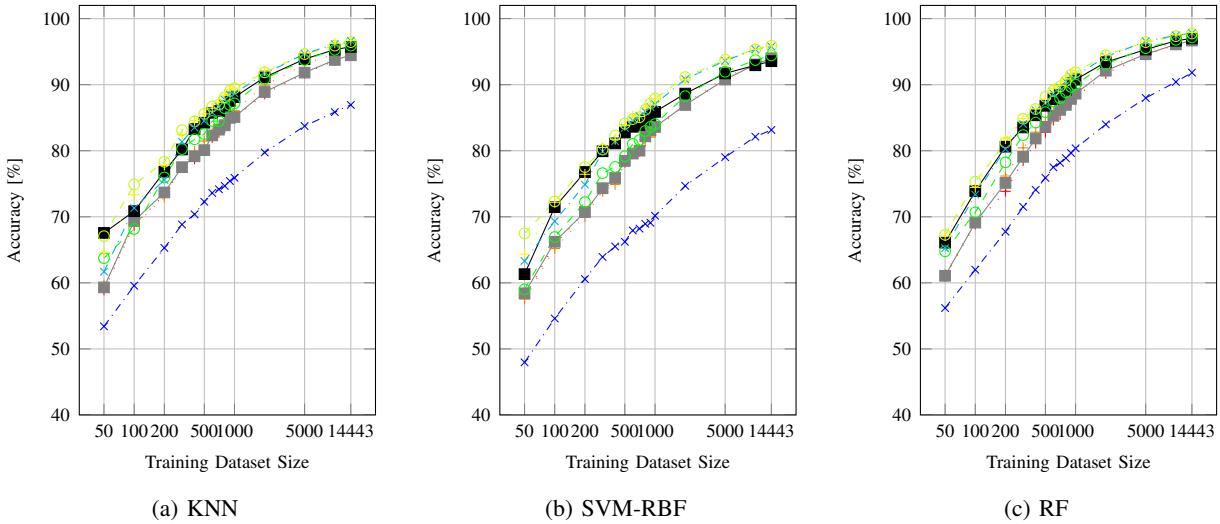


Fig. 8: Accuracy of KNN, SVM-RBF, RF with various training data set size (Number of Samples). The x-axis is in logarithmic scale.

decided to present changes in the Cipher suites and not in other cases since a number of works such as [50] have used the Cipher suite as important parameters in their classification system.

To determine the effect of the training data set size, we ran an experiment with various training set sizes (between 50 samples and a full data-set, $14,443 = 0.7 \times 20,633$ samples).

Figure 8 shows that although the training set had fewer samples, the system still achieved a reasonable accuracy of 80% for the baseline features and 85% when the new features were added. We then investigated whether our system would achieve high accuracy in the case of short session time which means fewer data in each session. To do so, in the next experiments we built a training and test set from our session using up

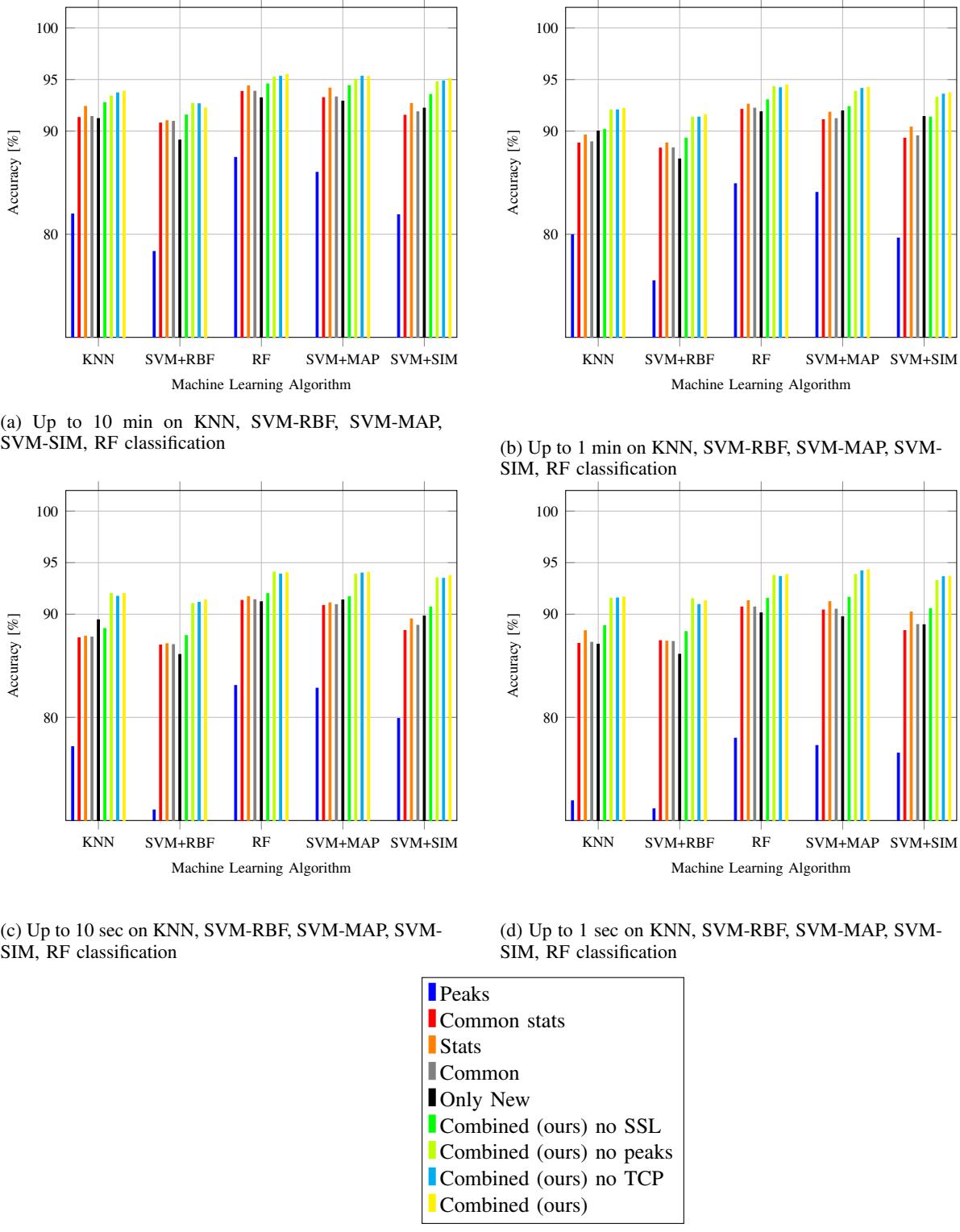


Fig. 9: Real time Tuple classification of short period of a session (1 second, 10 seconds, 1 minute and 10 minutes) using the KNN, SVM-RBF, SVM-MAP, SVM-SIM, RF classification algorithms

to X seconds/minutes (1 second, 10 seconds, 1 minutes, 10 minutes). Figure 9 shows that the accuracy decreased but was still high (close to 94%). Moreover, the effect of shortening the session slightly decrease after 1 minute of session time, where the results for 1 minute are comparable to the accuracy

of 1 second.

After observing that using short sessions did not have a pronounced effect on the results, we investigated the run time of training and testing. Figure 10 shows that the training time of our algorithms was between 10 seconds (RF) and 300

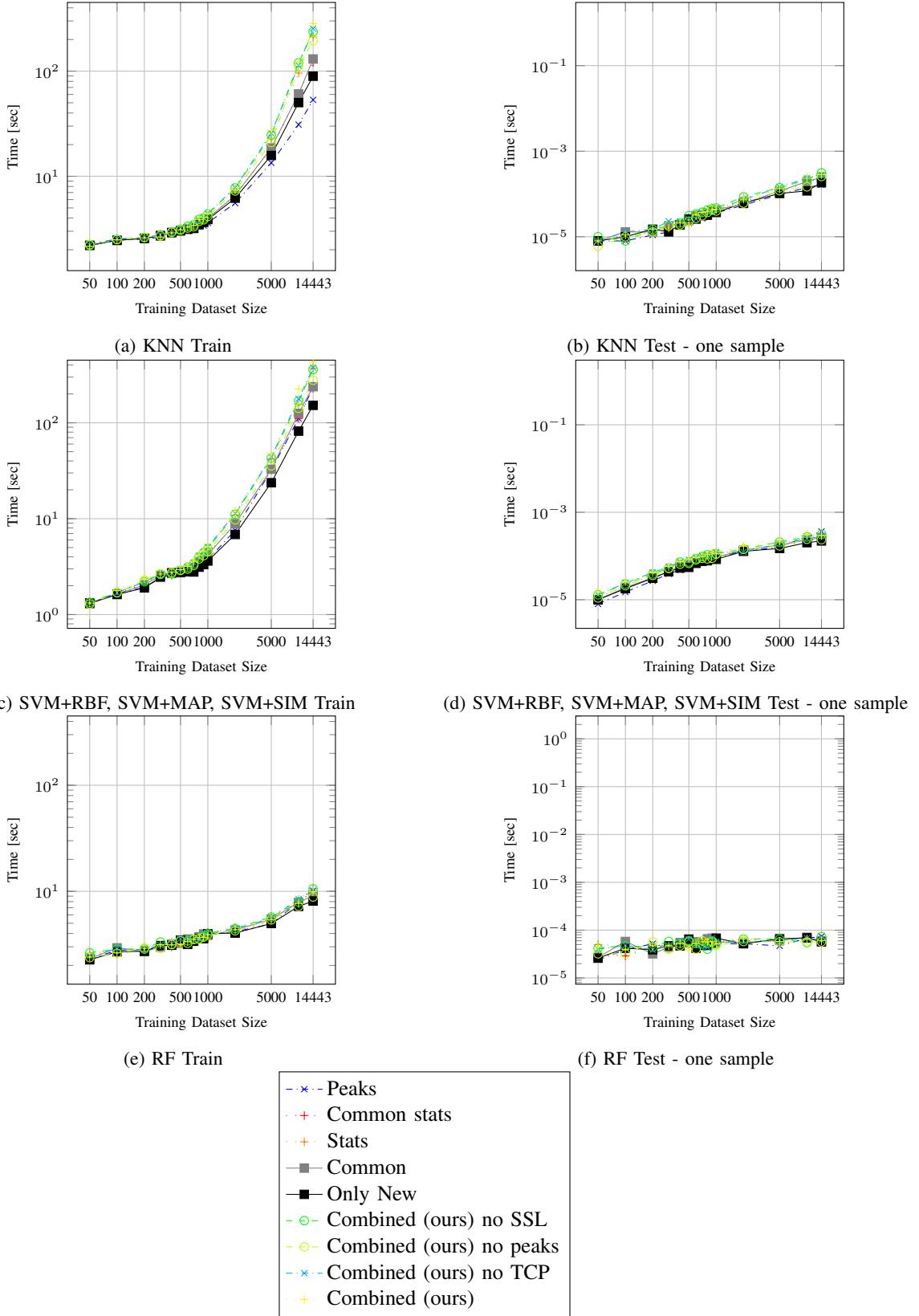


Fig. 10: Training (including cross validation) and testing time of the machine learning algorithms for various training dataset sizes. The x-axis is in logarithmic scale. The testing time for one sample (number of samples = 6190)

seconds (SVM) whereas the run time of the testing was less than 1 second, indicating that our testing algorithms can run

on real time networks.

V. POSSIBLE COUNTERMEASURES AND LIMITATIONS

Although users and service providers often assume that if they use the right encryption and authentication mechanisms their communications are secure, they are still vulnerable. As presented in this paper, it is possible to develop classifiers for TLS/SSL encrypted traffic that are able to discriminate between OSs, browsers and applications. We showed that changing protocol parameters such as cipher suites or using VPNs as counter-measures at test time reduced accuracy, but our system was able to identify the information with reasonable accuracy. While it is beyond the scope of the paper to investigate all possible countermeasures, we discuss some related issues.

Padding techniques are another simple countermeasure which may be effective against traffic analysis approaches. However, padding countermeasures are already standardized in TLS, explicitly to frustrate attacks on a protocol that are based on analysis of the lengths of exchanged messages [6]. The intuition is that the information is not hidden efficiently, and the analysis of these features may still allow analysis.

The main limitation of our approach has to do with our implementation of supervised learning algorithms. This technique is generally more efficient than unsupervised learning since it benefits from knowing each class of interest. However, it has three main drawbacks: (1) The training dataset has to be labeled by a human expert; (2) It is hard to recognize classes of events that have not been used during the training phase; (3) Upgrades (OS, browsers, applications) can change traffic patterns.

We mitigated the first limitation by using an automatic approach to label the network traces collected for the training phase. However, the second limitation cannot be addressed without revising the entire approach. In order to mitigate the third limitation we connected our lab to a VPN network that added another layer of encryption and we changed the Cipher suites number and values of the browsers. In both cases we used the same classifiers when classifying the operating system and the browser. Fig. 7 shows that in both cases although the results were slightly affected, the accuracy remained high.

VI. CONCLUSION AND FUTURE WORK

The framework proposed in this paper is able to classify encrypted network traffic and infer which operating system, browser and application the user is employing even in the case where a user tries to manipulate the traffic. We also showed that despite the use of SSL/TLS, our traffic analysis approach is an effective tool. An eavesdropper can easily leverage the information about the user to fit an optimal attack vector. An interesting extension of this work would be to add more user abilities and develop a system that will be robust to other types of attacks such as fake samples.

ACKNOWLEDGMENT

This research was supported by the InfoMedia consortium.

REFERENCES

- [1] R. Atterer, M. Wnuk, and A. Schmidt. Knowing the user's every move: User activity tracking for website usability evaluation and implicit interaction. In *Proceedings of the 15th International Conference on World Wide Web*, pages 203–212, 2006.
- [2] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing user navigation and interactions in online social networks. *Information Sciences*, 195:1 – 24, 2012.
- [3] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger. Understanding online social network usage from a network perspective. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, pages 35–48, 2009.
- [4] G. Ateniese, B. Hitaj, L. V. Mancini, N. Vincenzo Verde, and A. Villani. No place to hide that bytes won't reveal: Sniffing location-based encrypted traffic to track user position. *CoRR*, 2015.
- [5] Google. Google webmaster central blog: Https as a ranking signal, 2014.
- [6] E. Rescorla. The transport layer security (TLS) protocol version 1.3, Dec. 2015.
- [7] V.F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *1st IEEE European Symposium on Security and Privacy*, mar 2016.
- [8] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions On Information Forensics and Security*, 2016.
- [9] M. Crotti, F. Gringoli, P. Pelosato, and L. Salgarelli. A statistical approach to IP level classification of network traffic. In *International Conference on Communications*, pages 170–176, June 2006.
- [10] T. Okabe, T. Kitamura, and T. Shizuno. Statistical traffic identification method based on flow-level behavior for fair VoIP service. In *IEEE Workshop on VoIP Management and Security*, pages 35–40, April 2006.
- [11] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on ssh. In *Proceedings of the 10th Conference on USENIX Security Symposium*, pages 25–25, 2001.
- [12] B. Canvel, A. Hiltgen, S. Vaudenay, and M. Vuagnoux. Password interception in a ssl/tls channel. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 583–599. Springer Berlin Heidelberg, 2003.
- [13] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 5:1–5:16, 2007.
- [14] Y. Liu, C. Ou, Z. Li, C. Corbett, B. Mukherjee, and D. Ghosal. Wavelet-based traffic analysis for identifying video streams over broadband networks. In *IEEE Global Telecommunications Conference*, pages 1–6, Nov 2008.
- [15] Y. Liu, A-R Sadeghi, D. Ghosal, and B. Mukherjee. Video streaming forensic content identification with traffic snooping. In *Information Security*, volume 6531 of *Lecture Notes in Computer Science*, pages 129–135. Springer, 2011.
- [16] AM. White, AR. Matthews, KZ. Snow, and F. Monrose. Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2011.
- [17] CV. Wright, L. Ballard, F. Monrose, and GM. Masson. Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob? In *USENIX Security*, 2007.
- [18] A. Dainotti, A. Pescape, and KC. Claffy. Issues and future directions in traffic classification. *IEEE Network*, 2012.
- [19] V. Paxson. Empirically derived analytic models of wide-area tcp connections. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 2(4):316–336, 1994.
- [20] R. Alshammari and AN. Zincir-Heywood. Unveiling skype encrypted tunnels using gp. In *IEEE Congress on Evolutionary Computation (CEC)*, 2010.
- [21] S. Zander, T. Nguyen, and G. Armitage. Self-learning ip traffic classification based on statistical flow characteristics. In *6th International Workshop Passive and Active Network Measurement*, 2005.
- [22] I. Paredes-Oliva, I. Castell-Uroz, P. Barlet-Ros, X. Dimitropoulos, and J. Sole-Pareta. Practical anomaly detection based on classifying frequent traffic patterns. In *INFOCOM WKSHPS*, 2012.
- [23] D. Zhang, C. Zheng, H. Zhang, and H. Yu. Identification and analysis of skype peer-to-peer traffic. In *Fifth International Conference on Internet and Web Applications and Services (ICIW)*, 2010.
- [24] D. Bonfiglio, M. Mellia, M. Meo, and D. Rossi. Detailed analysis of skype traffic. *IEEE Transactions on Multimedia*, 11:117–127, 2009.

- [25] KT. Chen, CY. Huang, P. Huang, and CL. Lei. Quantifying skype user satisfaction. In *SIGCOMM*, Sep. 2006.
- [26] E. Hjelmvik and W. John. Statistical protocol identification with spid: Preliminary results. In *Swedish National Computer Networking Workshop*, 2009.
- [27] R. Bar-Yanai, M. Langberg, D. Peleg, and L. Roditty. Realtime classification for encrypted traffic. In *Experimental Algorithms*, 2010.
- [28] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia. Reviewing traffic classification. In *DataTraffic Monitoring and Analysis*, 2013.
- [29] Z. Cao, G. Xiong, Y. Zhao, Z. Li, and L. Guo. A survey on encrypted traffic classification. In *Applications and Techniques in Information Security*, 2014.
- [30] P. Barlet-Ros T. Bujlowa, V. Carela-Espaolb. Independent comparison of popular dpi tools for traffic classification. *Computer Networks*, 76:75–89, 2015.
- [31] T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys and Tutorials*, 10:56–76, 2008.
- [32] R. Bar Yanai, M. Langberg, D. Peleg, and L. Roditty. Realtime classification for encrypted traffic. In *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 373–385. Springer Berlin Heidelberg, 2010.
- [33] M. Calisti, S. Meer, and J. Strassner. *Advanced Autonomic Networking and Communication (Whitestein Series in Software Agent Technologies and Autonomic Computing)*. Birkhäuser Basel, 1 edition, 2008.
- [34] M. Korczynski and A. Duda. Classifying service flows in the encrypted skype traffic. In *IEEE International Conference on Communications (ICC)*, pages 1064–1068, June 2015.
- [35] D. Zhang, C. Zheng, H. Zhang, and H. Yu. Identification and analysis of skype peer-to-peer traffic. In *Internet and Web Applications and Services*, pages 200–206, May 2010.
- [36] A. W. Moore and K. Papagiannaki. Toward the accurate identification of network applications. In *Proceedings of the 6th International Conference on Passive and Active Network Measurement*, pages 41–54, 2005.
- [37] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 50–60, 2005.
- [38] Nigel Williams, Sebastian Zander, and Grenville Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *SIGCOMM Comput. Commun. Rev.*, 36(5):5–16, Oct 2006.
- [39] R. Alshammari and A.N. Zincir-Heywood. Unveiling skype encrypted tunnels using gp. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, July 2010.
- [40] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. In Chadi Barakat and Ian Pratt, editors, *Passive and Active Network Measurement*, volume 3015 of *Lecture Notes in Computer Science*, pages 205–214. Springer Berlin Heidelberg, 2004.
- [41] B. Niemczyk and P.Rao. Identification over encrypted channels. In *blackHat*, Aug. 2014.
- [42] W. de Donato, A. Pescap, and A. Dainotti. Traffic Identification Engine: An Open Platform for Traffic Classification. *IEEE Network*, 28(2):56–64, Mar 2014.
- [43] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru. How secure and quick is quic? provable security and performance analyses. In *Symposium on Security and Privacy (SP)*, pages 214–231, May 2015.
- [44] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. The most dangerous code in the world: Validating ssl certificates in non-browser software. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 38–49, 2012.
- [45] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail. In *IEEE Symposium on Security and Privacy (SP)*, pages 332–346, 2012.
- [46] H. Shi, H. Li, D. Zhang, C. Cheng, and W. Wu. Efficient and robust feature extraction and selection for traffic classification. *Computer Networks*, 119:1 – 16, 2017.
- [47] L. Huang, A. D. Joseph, B. Nelson, B. I.P. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, pages 43–58, 2011.
- [48] M. Zalewski. P0f. <http://lcamtuf.coredump.cx/p0f3/>. Accessed: 2016-11-12.
- [49] B. Anderson and D. A. McGrew. OS fingerprinting: New techniques and a study of information gain and obfuscation. *CoRR*, abs/1706.08003, 2017.
- [50] M. Husák, M. Čermák, T. Jirsík, and P. Čeleďa. Https traffic analysis and client identification using passive ssl/tls fingerprinting. *EURASIP Journal on Information Security*, 2016(1):1–14, 2016.
- [51] A. Dvir, J. Muehlstein, Y. Zion, R. Dubin, and O. Pele. The research dataset. <http://www.ariel.ac.il/sites/amitd/data-sets>.
- [52] A. Dvir, J. Muehlstein, Y. Zion, R. Dubin, and O. Pele. The crawler code. <https://github.com/yzion/infomedia-crawler>.
- [53] A. Dvir, J. Muehlstein, Y. Zion, R. Dubin, and O. Pele. The feature extraction code. <https://github.com/JonMuehlste/pcap-feature-extractor>.
- [54] A. Dvir, J. Muehlstein, Y. Zion, R. Dubin, and O. Pele. The machine learning algorithm code. <https://github.com/JonMuehlste/ml-score>.
- [55] R. Dubin, A. Dvir, O. Pele, J. Muehlstein, Y. Zion, M. Bahumi, and I. Kirshenboim. Analyzing https encrypted traffic to identify users operating system, browser and application. In *IEEE Consumer Communications and Networking Conference*. IEEE, Jun. 2017.
- [56] I. Paredes-Oliva, I. Castell-Uroz, P. Barlet-Ros, X. Dimitropoulos, and J. Sole-Pareta. Practical anomaly detection based on classifying frequent traffic patterns. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 49–54, March 2012.
- [57] K. T. Chen, C. Y. Huang, P. Huang, and C. L. Lei. Quantifying skype user satisfaction. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 399–410, 2006.
- [58] D. Bonfiglio, M. Mellia, M. Meo, and D. Rossi. Detailed analysis of skype traffic. *IEEE Transactions on Multimedia*, 11(1):117–127, Jan 2009.
- [59] E. Hjelmvik and W. John. Statistical protocol identification with spid: Preliminary results. In *Swedish National Computer Networking Workshop*, pages 399–410, 2009.
- [60] R. Bar Yanai, M. Langberg, D. Peleg, and L. Roditty. Realtime classification for encrypted traffic. In Paola Festa, editor, *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 373–385. Springer Berlin Heidelberg, 2010.
- [61] 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*, pages 255–263, 2006.
- [62] 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*, pages 103–114, 2011.
- [63] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th conference on USENIX Security Symposium*, pages 1–17, 2004.
- [64] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 605–616, 2012.
- [65] X. Luo, P. Zhou, E. W. W. Chan, W. Lee, R. K. C. Chang, and R. Perdisci. Https: Sealing information leaks with browser-side obfuscation of encrypted flows. In *Network and Distributed Systems Symposium (NDSS)*, 2011.
- [66] X. Zhou, S. Demetriadou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and K. Nahrstedt. Identity, location, disease and more: Inferring your secrets from android public resources. In *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security*, pages 1017–1028, 2013.
- [67] W. Enck, P. Gilbert, B-G Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, pages 393–407, 2010.
- [68] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, pages 281–287, 2010.
- [69] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic. Who do you sync you are?: Smartphone fingerprinting via application behaviour. In *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 7–12, 2013.
- [70] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos. Profiledroid: Multi-layer profiling of android applications. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, pages 137–148, 2012.
- [71] L. Sinha, S. Bhandari, P. Faruki, M. S. Gaur, V. Laxmi, and M. Conti. Flowmine: Android app analysis via data flow. In *Consumer Communications & Networking Conference (CCNC), 13th IEEE Annual*, pages 435–441. IEEE, 2016.

- [72] H. Fereidooni, V. Moonsamy, M. Conti, and L. Batina. Efficient classification of android malware in the wild using robust static features. *Protecting Mobile Networks and Devices: Challenges and Solutions*, page 181, 2016.
- [73] B. Saltformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic. In *10th USENIX Workshop on Offensive Technologies (WOOT)*, Aug 2016.
- [74] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pages 297–304, 2015.
- [75] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Robust smart-phone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security*, 13(1):63–78, Jan 2018.
- [76] NetRESec. Passive os fingerprinting. <http://www.netresec.com/?page=Blog&month=2011-11&post=Passive-OS-Fingerprinting>. Accessed: 2016-11-12.
- [77] P. Matousek, O. Rysavy, M. Gregrand, and M. Vymaltil. Towards identification of operating systems from the internet traffic. In *5th International Conference on Data Communication Networking*, pages 21–27, July 2014.
- [78] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS '06*, pages 16–25, 2006.
- [79] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE Transactions on Dependable and Secure Computing*, 2017.
- [80] Selenium. Selenium automates browsers. <http://www.seleniumhq.org/>. Accessed: 2016-02-28.
- [81] SplitCap. Splitcap - pcap file splitter. <http://www.netresec.com/?page=SplitCap>. Accessed: 2016-02-28.
- [82] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer, Berlin, 2001.
- [83] R. Dubin, A. Dvir, O. Pele, and O. Hadar. I know what you saw last minute-encrypted http adaptive video streaming title classification. In *Black Hat*, 2016.
- [84] P. Ameigeiras, J. Ramos-Muoz, J. Navarro-Ortiz, and J. M. Lpez-Soler. Analysis and modelling of youtube traffic. *Transactions on Emerging Telecommunications Technologies*, 23(4):360–377, 2012.
- [85] A. Rao, A. Legout, Y.S. Lim, D. Towsley, C. Barakat, and W. Dabbous. Network characteristics of video streaming traffic. In *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2011.
- [86] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.
- [87] C. Cortes and V. Vapnik. Support vector machine. *Machine learning*, 1995.
- [88] C-C. Chang and C-J Lin. LIBSVM: A library for support vector machines. *IST*, 2011.
- [89] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti. Similarity-based classification: concepts and algorithms. *The Journal of Machine Learning Research*, 10:747–776, 2009.
- [90] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [91] T. K. Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

VII. BIOGRAPHY



Dr. Amit Dvir received the B.Sc., M.Sc., and Ph.D. degrees from Ben-Gurion University, Beer Sheva, Israel, all in communication systems engineering. He is currently a Faculty Member with the Computer Science Department and the Cyber Center, Ariel University, Israel. From 2011 to 2012, he was a Post-Doctoral Fellow with the Laboratory of

Cryptography and System Security, Budapest, Hungary. His research interests include enrichment data from encrypted traffic.



big-data, cyber security

Mr. Yehonatan Zion received BSc and MSc degrees from Ariel University, Israel, in 2016 and 2018, respectively, all in computer science. Yehonatan is now working as a Research and Software engineer in FlashNetworks in the field of big-data and mobile network communication.



Mr. Jonathan Muehlstein received BSc and MSc degrees from Ariel University, Israel, in 2016 and 2018, respectively, all in computer science. Yehonatan is now working as a Research and Software engineer in Viaccess-Orca in the field of big-data and recommendation systems.



Dr. Ofir Pele received the B.Sc. degree in computer science and life science and the M.Sc. and Ph.D. degrees in computer science from the Hebrew University of Jerusalem, Israel, in 2003, 2005, and 2011, respectively. From 2011 to 2013, he was a PostDoctoral Fellow with the Department of Computer and Information Science, University of Pennsylvania, working with the late Prof. B. Taskar.

In 2013, he joined Ariel University as a Lecturer (Assistant Professor). His work focuses on novel algorithms that combine practical applicability and theoretical insights. He strongly believes in reproducible research and publishing open source code of his work. His work has been successfully used by several research groups, including groups from prestigious universities and research centers. His research interests include machine learning, computer vision, and their applications.



Dr. Ran Dubin received the BSc, the MSc (cum laude) and the PhD degrees from the Ben-Gurion University of the Negev, Israel, in 2010, 2013, and 2016, respectively, all in communication system engineering. Ran was assigned by the Israeli district courts to consult on

the subject of Internet network class claims issues. Ran's previous industry experience was developing machine learning and big-data algorithms in the cyber protection and network communication fields