I. J. Computer Network and Information Security, 2015, 7, 10-23
Published Online June 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2015.07.02

A Model for Detecting Tor Encrypted Traffic using Supervised Machine Learning

Alaeddin Almubayed
Yahoo Inc., California, US
Email: alaa_bus@yahoo.com

Ali Hadi and Jalal Atoum
Princess Sumaya University for Technology (PSUT), Amman, Jordan
Email: {a.hadi, atoum}@psut.edu.jo

Abstract—Tor is the low-latency anonymity tool and one of the prevalent used open source anonymity tools for anonymizing TCP traffic on the Internet used by around 500,000 people every day. Tor protects user's privacy against surveillance and censorship by making it extremely difficult for an observer to correlate visited websites in the Internet with the real physical-world identity. Tor accomplished that by ensuring adequate protection of Tor traffic against traffic analysis and feature extraction techniques. Further, Tor ensures anti-website fingerprinting by implementing different defences like TLS encryption, padding, and packet relaying. However, in this paper, an analysis has been performed against Tor from a local observer in order to bypass Tor protections; the method consists of a feature extraction from a local network dataset. Analysis shows that it's still possible for a local observer to fingerprint top monitored sites on Alexa and Tor traffic can be classified amongst other HTTPS traffic in the network despite the use of Tor's protections. In the experiment, several supervised machine-learning algorithms have been employed. The attack assumes a local observer sitting on a local network fingerprinting top 100 sites on Alexa; results gave an improvement amongst previous results by achieving an accuracy of 99.64% and 0.01% false positive.

Index Terms—Anonymity, Censorship, Interception, Machine Learning, Tor, Traffic Analysis, Traffic Classification

I. Introduction

Tor is widely known low latency network anonymity project and is currently used by around 500,000 daily users and carrying 2500 MB of data per second [1]. Tor stands for "The onion router" or the onion routing network, it provides two ways bidirectional anonymized connection over the network. Tor provides strong implementation, which protects against both sniffing and analysis making a secure communication to protect both data confidentiality and users privacy. TLS protocol is used in Tor communication to provide the required encryption [2].

For example, if we have both Bob and Alice communicating on a public Internet connection, by using the mean of Tor, they can ensure that their communication cannot be intercepted or monitored by eavesdroppers and that the information passed back and forth is encrypted and anonymized.

Tor is free open source software that works almost on every platform, once Tor installed, users can use web browser to anonymize their traffic. Traffic passes between Tor nodes and users are secure via strong encryption [3]. Moreover, Tor works perfectly on modern browsers such as Firefox and Chrome with Tor bundles.

Bundles enable users to install Tor as browser extension that makes it easier for users to protect their communication and attain anonymity and privacy [4]. However, despite Tor is used for online anonymity, it's heavily used by hackers and cybercriminals in order to avoid traceability [5]. With the increasing usage of the Internet, concerns over censorship and privacy have become a big goal, users heavily rely on anonymity tools in order to conceal their identity and gain privacy. For those users, anonymity is significantly important and Tor analysis against various attacks is deemed necessary to ensure adequate protection of user's privacy.

Further, although there is a huge evolution of developing more anonymity tools, blocking anonymous traffic and developing anti-blocking tools attracting many researchers [6], this makes a strong reason for Tor to monitor and track down anti-anonymity tools to ensure secure anonymity for users all the times. In fact, the detection of anonymity tool is become a hot topic as there is an infinite battle between developers work to improve the anonymity tools and organizations, governments who work also tremendously to break anonymity. Internet users strongly believe that the need for anonymity to protect user's privacy is very important; users in totalitarian regimes strongly rely on such networks to freely communicate. Breaking Tor anonymity in fact reduces the protection that Tor claims to have for concealing users identities, and thus, increases the chances for those totalitarian regimes to physically identify users, which could lead to severe consequences such as imprisoning or even life threatening [7].

In this paper, the research has considered many machine learning (ML) algorithms in order to fingerprint Tor usage in the network. This study will help Tor developers to improve Tor security, provide more advanced techniques, and solutions in order to boost Tor anonymity. Furthermore attain a complete protection for the users, this in case the same analysis has conducted by either attacker or totalitarian regime. The main objectives of this work can be summarized as the following:

1. Researching different techniques and tools in order to identify Tor usage in the network by tracing an offline network traffic data.
2. Researching the possibility of fingerprinting Tor traffic of top 5 sites on Alexa amongst other top 100 sites on Alexa using ML algorithms by extracting statistics in the SSL flows used by Tor software.
3. Generating an extensive HTTPS traffic along with Tor traffic using two virtual machines (VMs).
4. Feature selection exercise from network pcap files generated from a different network traces to build the ML data model.
5. Conducting an analysis on how many packets of SSL flows are required to classify Tor amongst HTTPS.
6. Performing a detailed experimentation to measure the accuracy of ML classifiers.

In this research, studying the possibility of identifying the individual users who use Tor is out of this research scope; the focus is to only identifying Tor usage in an offline network traces via websites fingerprinting. Also studying ML algorithms in this research is limited, since this is more of computer science knowledge, the focus is mainly on researching traffic classification for Tor using specific ML algorithms in order to perform websites fingerprinting. Further, the analysis of Tor is conducted in a closed-world local network environment considering the fact that it's difficult to obtain traffic from an open-world environment such as Internet Service Providers (ISP).

II. Tor Background

Tor allows people to access and publish content on the Internet without being tracked or identified or cleared to authorities. Considering the usage of Tor by various and different type of people the risk is varied from a risk of child accessing forbidden sites to other type of risks such as employees or political activist accessing Tor where the risk is higher. However, while many people agree on positive reasons to use Tor, some people see Tor as a big threat that could make criminals to commit their crimes with impunity. The good reasons of using Tor are several, for example normal people use Tor to protect their information from external adversaries, and also, military uses Tor to protect government communication, in addition to that, law enforcement offices and agencies are using Tor for their investigation and operation. Low and high profile people also use Tor to make an opinion that may be unpopular or conflict with their public persona [8].

Tor completely relies on TLS protocols for its network communication. TLS encrypts and authenticates the communication between Tor instances.

A. Transport Layer Security

Netscape Communication Corporation first introduced secure Socket Layer (SSL) protocol in 1995 to enable e-commerce transaction security on the web. TLS is being used heavily nowadays by most Internet communication to protect confidentiality through encryption and integrity, as well as authentication, to ensure a safe transaction. However, to achieve this, SSL protocol was built up over the application layer directly on the top of TCP, which enables the protocol to work on HTTP, SMTP, FTP, and many others. The primary reason of SSL and TLS is to protect HTTP traffic in the network. In HTTP, when a new TCP connection is created, the client sends the request to the server and then the server responds back with the content, when SSL is utilized, the client first create a TCP connection and establishes an SSL stream channel to relay the TCP connection, at that point of time, the HTTP request is sent over the SSL connection instead of the regular TCP connection. SSL and TLS handshake cannot be understood by the ordinary HTTP, thus, a protocol specification HTTPS is used instead to indicate the use of a connection over SSL [9]

TLS is layered protocol and consists of mainly two layer protocols, at the lower level is the record protocol which is responsible for transmitting the message, fragments the data into blocks, and many other steps. On the top layer is the Handshake Protocol, Alert Protocol, Change Cipher Spec Protocol and Application Protocol, Fig. 1, which shows TLS protocol, layers. TLS handshake protocol allows both client and server to authenticate and exchange encryption keys and algorithm before the protocol starts to send data over the network [10].

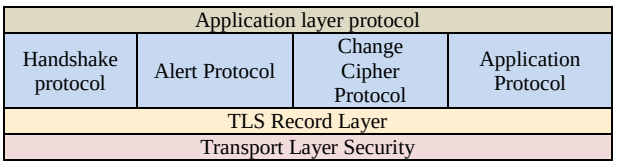

Fig. 1. TLS Protocol Layers

B. Onion Routing

The Onion Router (OR) was original created for Sun Solaris 2.4 in 1997, which include proxies for remote logins, email, and web browsing, also file transfer protocol (FTP) [11]. The main purpose of onion routing is to provide a real-time bidirectional anonymous interaction between two parties that is resistant to eavesdropping, sniffing and traffic analysis. Onion routing consists of a series of ORs connected in a way that each OR has a dedicated socket connection to a set of neighboring ones. However, to build up the anonymous connection, the application initiates a series of connections to a set of Onion Router Proxies (ORPs) that ultimately build up the anonymous connection. The routing occurs at the application layer of the protocol stack, and not on the IP

layer. However, the IP network is the one who determines where should the data move between each individual onion routers.

III. Related Work

Tor achieves anonymity by make it very difficult for an adversary to identify client and server identities. In Tor design, the entry node only knows the client who communicates with middle node, and the middle node knows the entry node is communicating with another machine exit node. The middle relay machine cannot tell if it's the middle node in the circuit or not. Also the exit relay knows the middle node, which communicates with the server (target destination). Finally, the server believes the connection is coming from the exit node [12]. Historically, an extensive number of work on attacking Tor anonymity circuits, which can degrade the anonymous communication over Tor; most of these attacks are based on traffic analysis. However, attacks based on traffic analysis may suffer high rate of false positives (FP) due to a number of reasons, such as Internet traffic dynamics and determining the required number of packets for the statistical analysis of traffic. That said, timing and latency are important metrics in traffic analysis to identify Tor as well as packet counting and volume metrics [13].

A previous work on path selection focused on latency as property link and take delay in account primarily.

However in this attack by [14], attacker assumes different approach, which is identifying the important of latency as indicator of congestion, and accordingly, suggesting an improved path selection algorithm. Further, Tao proposed a way for Tor clients to respond to short-term congestion by building timeout mechanism.

Existing traffic analysis attacks against anonymous communication can be classified into two main categories: traffic confirmation attacks and traffic analysis attacks. Each category consists of both passive and active attacks. Passive traffic analysis techniques is when the adversary records the traffic passively and identify the resemblance between client inbound traffic and server outbound traffic. Meanwhile, the active attack, aims to embed specific secret signal (or marks) into the target traffic and detect it [15].

Meanwhile, traffic confirmation attack is when an adversary tries to confirm that two parties are communicating with each other over Tor by observing patterns in the traffic, such as timing and volume of the traffic. Ideally, traffic confirmation attacks are not in the focus of Tor's threat model. Instead, Tor increases the focus on preventing traffic analysis attacks, this occurs when adversary tries to determine in which points in the network a traffic pattern based attack should be executed [15].

IV. Tor Fingerprint Methodology

The goal of this research is to fingerprint Tor traffic flows in a local network environment in order to break Tor's anonymity and identify top monitored sites on Alexa using ML classification techniques. There are several steps involved in Tor fingerprinting attack within a local network environment; local network environment means two things. First, all web pages are known in advance, and second, the attack is launched by a local attacker. The attacker observes the encrypted traffic to find conclusions from certain features in the traffic such as packet sizes, volume of data transferred, timing and many others. This type of attack is considered in this research to ensure the comparability of the outcome results to related works. This method however is not in the position of breaking the cryptographic used in Tor, although it does not provide messages semantic, it can provide a way to observe specific patterns in order to reveal a known traffic instances like web pages.

In real world scenario, if a user runs Tor OP in a shared local network, other users sit on the same network may use different applications, and thus, passing different type of network traffic traces such as HTTP, HTTPS, FTP and others. Tor's uses TLS encryption between client, ORs and destination server, thus, the hypothesis is that the traffic of Tor should have similar characteristic as any other TLS traffic such as HTTPS. Yet, if variation in the traffic characteristic is found, then Tor instances can be fingerprinted amongst other TLS traffic and anonymity can be broken. In the experiment, HTTPS encrypted traffic is considered as majority of sites encrypts their communication use TLS encryption over port 443. Similarly, Tor traffic is considered from a user (victim) uses Tor OP on the same local network browsing top 5 sites on Alexa. Therefore, by identifying the variations in the traffic instances between the HTTPS traffic and Tor's victim traffic in the same local network environment is the goal for this study.

In order to find those variations in the traffic characteristic between Tor and HTTPS, ML methods need to be employed using statistical classification technique [16]. The focuses on using ML methods to detect patterns in the packet information is to extrapolate and predict traffic type contained within a TLS flow, which in this research, using Tor encrypted traffic data and HTTPS data to train the system.

Generally, the Tor Fingerprinting Methodology steps in this work can be summarized in Fig. 2, as follows:

- Step-1, data collection step, it's required to capture a ground truth, or HTTPS data for which the underlying application is known. At the same time, collects Tor traffic instances of top 5 sites on Alexa to represents Tor instances. The data collected is to be used to train the model using ML methods.
- Step-2 is feature extraction and feature selection; feature extraction is crucial in order to detect the variation between Tor instances and HTTPS instances and feature selection is required to identify which features to be used that improve the accuracy of web sites fingerprinting.
- Step-3, labeling process means marking each row in the traffic with the corresponded label.

- Step-4, classifying traffic flows based on those characteristics variations either as Tor's site or regular HTTPS traffic. In the following subsections are the details descriptions of each step.

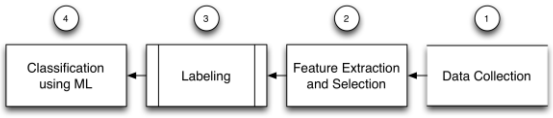

Fig. 2. Methodology steps for fingerprinting Tor

A. Data Collection

To validate the method used in this research, there is a need for a ground truth, or SSL connections for which the underlying application is known. Therefore HTTPS traffic data is required for building the training dataset. Although there is no public dataset sources that can be used in the experiment, similar technique has been considered for data generation from [14] which previously known to achieve higher accuracy results ignoring the removal of SENDMEs as it did not affect the results that much. Precautions need to be taken in order to collect the data in the same way a realistic attacker would. Firefox browser and Selenium [17] Web Driver have been used to perform an automation browsing process, web sites used are taken from top 100 sites on Alexa in order to mimic the actual real user behavior on the local network environment.

Ideally, capturing those traffic traces can be accomplished from more than one machine; those captures consist of a raw data that is transmitted over the physical wire or wireless network at a given point, see Fig. 3. Each machine runs different encrypted services. Few machines run HTTPS traffic and others run Tor application to generate Tor encrypted traffic. In the experiment two virtual machines are used as clients, below is a details about the software stack used for that.

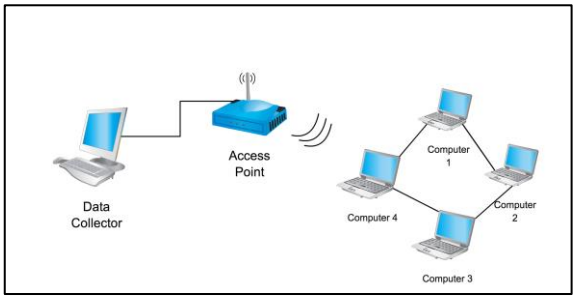

Fig. 3. Data collection

1) Environment Setup

In the data generation method, two virtual machines (VM) are used in order to generate the traffic for the experiment. The VM is a piece of software implementation of a computing environment in which an Operating System (OS) can be installed and run on an emulated physical computing environment, it basically utilizes the underlying physical hardware, including CPU, memory, hard disk, network and other hardware resources to create a virtual computing environment. Although resources of guest OSs and programs are running on virtualized computer, they are not aware that they are running on a virtual platform [18]. Also, in the study, different OS distribution systems have been installed to ensure emulating the actual traffic in the network, a breakdown of the OSs used as VMs to capture the network traffic is presented in Table I, each of these guests operating systems runs with specific VM configuration, a 512 memory RAM, and 20 GB of disk space with shared networking Network Address Translation (NAT) setup. Further, a distribution of Linux OSs on VMs with different processor architecture is used.

Table 1. Break Down Of Client Virtual Machines Operating Systems

Client #	***Operating System***	***Operating System Version***	***processor***
CVM1	Linux Ubuntu	12.04	64 bit
CVM2	Linux Backtrack	5.01.3	32 bit

Linux BackTrack is a Linux-based penetration-testing arsenal that aids security professionals in the ability to perform assessments in a purely native environment dedicated to hacking. Linux Ubuntu is the standard Linux distribution system powers millions of desktop PCs, laptops and servers around the world. Moreover, OSX machine in Table II is used for conducting the analysis.

Table 2. Analysis Machine

Client #	***Operating System***	***Operating System Version***	***processor***
A1	OSX	10.9.2	64-bit

2) Traffic Generation Tools

In order to obtain traffic traces for the dataset, capturing the data from VMs and use it to build training data sets is the first step, aforementioned VMs and Sniffer software were used to sniff and capture the traffic from A1. VMs are configured to run as NAT to A1 machine, which means traffic will always route through A1 machine, this provides two major benefits, first a full control on capturing the dataset, and second, control specific filters based on particular parameters without any traffic disruption that could affect the quality of the dataset which could cause invalid training data set. Ideally, there are many sniffers available in the market today, the most famous two are wireshark and tcpdump, however, any sniffer will suffice for the testing, but simple, flexible, low-cost, and fast tool is best, tcpdump works really well as sniffer for the experiment.

Tcpdump is a free open source sniffer, which uses libpcap to capture traffic and provides information about IP layer packets i.e. the length of the packet, the time the packet was sent or received, the order in which the packets were sent and received. Tcpdump is quite flexible and fast, it runs on most Linux and Unix variants, in fact, it's installed by default on many Linux distributions, and it has been ported to windows as WinDump. It does support variety of filters, with a powerful language for specifying individual filter types [19]. Further many other services are running on the VMs, Table III breakdowns the

services installed on each machine, with the corresponding versions, each one of these machines runs completely independent in its VM environment.

Table 3. Services running on the machines

Machines	*Services*
CVM1	Firefox 14.0.1 Tor 0.2.4.22 Netmate 0.9.5 Tcpdump 4.3.0 Libpcap 1.3.0
CVM2	Firefox 14.0.1 Tor 0.2.4.22 Netmate 0.9.5 Tcpdump 4.3.0
A1	Weka 3.7.3 Wireshark 1.10.7

The details about each VMs used is as the following:

a) CVM1

This VM is used to run Firefox and browse sites run over HTTPS. The traffic generated is intended to represent regular HTTPS traffic for the top 100 sites on Alexa. In the real world scenario, most of this traffic is generated during regular secure browsing activities such as email communication, social network sites, and financial activities. Unfortunately, these activities are somehow difficult to mimic. Thus, the approach taken in this thesis involves using Selenium [17] for automating web applications for testing purposes with a complete list of sites that run over HTTPS. Selenium operates by controlling a standard browser. This is important because the traffic generated needs to look like if it was captured by a user browsing the web doing his regular business activities. However, similar to Ian and Tao method [14] the method obtained a specific list of websites in a local network environment, those sites are taken from the top 100 sites on Alexa.com. Alexa is the leading provider of free, global web metrics ranks the top sites based on the number of unique users, page views, and number of visits [20].

b) CVM2

This VM runs a specific list of what expected to be the top monitored websites on Alexa, but this time with Tor OP, in the attack scenario, the expectation is that the victim uses this machine to browse top sites on Alexa. The same method of CVM1 is used in CVM2. The 5 sites used in the experiment are Google, Yahoo, Facebook, Wikipedia and Twitter.

3) Dumping Traffic

Dumping traffic is required in order to capture training datasets and record information to be used later for the classification part. Basically the A1 machine is used to capture all the traffic from all the CVM (i) machines, a sniffer is installed on the interface to capture the traffic passes through. Tcpdump is used to generate the packet capture (pcap) file which previously developed by wireshark. In the process, packets are captured and stored on A1 machine for further analysis using ML. Traffic generation process has been scheduled from each machine using a small bash script to record traffic on hourly basis. Tcpdump sniffer is installed in a way so it can capture traffic from two machines on a scheduled basis see Fig. 4.

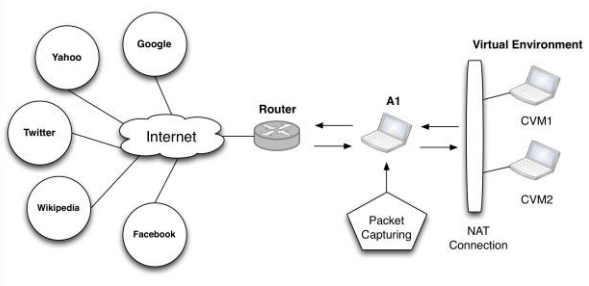

Fig. 4. Traffic capturing through A1

Traffic passes from CVM (i) through the NAT connection to websites servers. Since traffic scheduled to pass on a specific timeframe, HTTPS traffic was first generated from CVM1, and is called regular-HTTPS.pcap.

Similarly, traffic from the other CVM2 which runs Tor OP is captured, files named based on site corresponded to that traffic, example for Google traces, it's called Google-Tor.pcap and Yahoo traces Yahoo-Tor.pcap. The data generation took two weeks to finish and the final output files in a pcap format are listed in Table IV. The table contains the number of packets, flows along with the sizes for each. Fig. 5 shows a summery chart of each flow. After dumping the network traffic from CVM (i), the next step is to use those files to build the training model for the classification method. The traffic generated contains a number of flows; those flows will be used to create the model.

Table 4. Traffic and Their associated number of flows

Traffic Type	*Size / MB*	*Number of Packets*	*Number of Flows*	*Avg Packet Size / Byte*
HTTPS	808.7	1054835	38845	750.617
Google Tor	110.1	146151	5231	737.407
Yahoo Tor	155.4	206998	7959	734.596
Facebook Tor	132.7	160491	4085	810.785
Twitter Tor	132.2	171935	5577	752.653
Wikipedia Tor	87.7	122708	4465	698.716

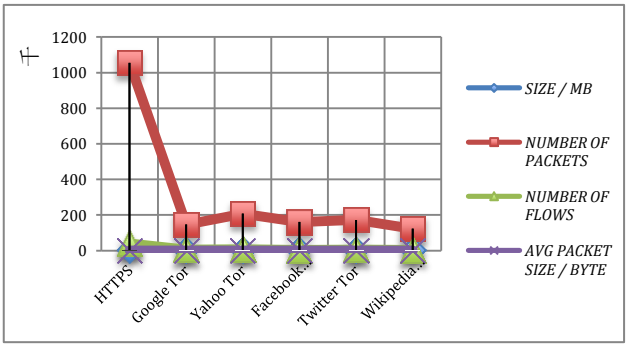

Fig. 5. Summery chart for all pcap files

B. Feature Extraction

In the experiment, in order to perform the fingerprinting attack, the dataset (or features) that represents each traffic type (Tor or HTTPS) needs to be extracted from the network dump file in order to use those features for our classification model to find characteristic variation between those instances.

The features need to be extracted from the network generated traffic *.pcap files, but first it's important to bring all the data together into a set of instances. In order to accomplish this, NetMate is used, NetMate is a traffic-monitoring tool, which converts IP packets into bi-directional flows and generates several statistics regarding these flows. The flows are actually defined using a sequence of packets, source IP address, destination IP address, source port, destination port, and type of protocol [21]. NetMate has been used to extract features as flow attributes from the traffic, NetMate works by processing the datasets, generating flows, and computing feature values which can be used to build the model, each flow is described by a set of statistical features and associated feature values.

1) Feature Selection

In total, 40 features were obtained from NetMate as of Table V, ignoring the other features including the protocol feature, which represent as (TCP=6 & UDP=17) considering that they don't impact the classification results positively or negatively [22] and proofed in this experiment. Further, It is important to mention that only TCP and UDP flows are considered, and specifically, flows that have at least one packet in each direction, and transport no less than one byte of payload. Also, there are a number of features have been excluded, IP addresses, and source/destination ports numbers to ensure that the results are not dependent from those biases.

2) Generating The Attribute Relationship File Format

Attribute relation file format (ARFF) is an input ASCII text file format that describes a list of instances sharing a set of attributes; it was developed by the ML Project at the Department of Computer Science of The University of Waikato to be used for machine learning software [23]. ARFF file has three main sections, RELATION, ATTRIBUTE and DATA. The header contains the relation declaration and an attribute declaration, RELATION is a string defined in the first line, ATTRIBUTE contains both name and data type, whilst DATA is the actual data declaration and actual instances line.

C. Labeling

Obtaining flows from network traffic using NetMate generates rows of attributes separated by commas in ARFF file format. Those values will be used to build the training dataset model using Weka, which is a collection of ML algorithms for data mining tasks [23]. In order to train the system Weka to use supervised ML with Weka defaults to validate the method. There is a need for a truth, or SSL connections for which the underlying application is known. In other words, there is a need to

Table 5. Features obtained from Netmate

No#	*Abbreviations*	*Features Description*
1	Dscp	The protocol (ie. TCP = 6, UDP = 17)
2	total_fpackets	Total packets in the forward direction
3	total_fvolume	Total bytes in the forward direction
4	total_bpackets	Total packets in the backward direction
5	total_bvolume	Total bytes in the backward direction
6	min_fpktl	The size of the smallest packet sent in the forward direction (in bytes)
7	min_fpktl	The mean size of packets sent in the forward direction (in bytes)
8	min_fpktl	The size of the largest packet sent in the forward direction (in bytes)
9	std_fpktl	The standard deviation from the mean of the packets sent in the forward direction (in bytes)
10	min_bpktl	The size of the smallest packet sent in the backward direction (in bytes)
11	mean_bpktl	The mean size of packets sent in the backward direction (in bytes)
12	max_bpktl	The size of the largest packet sent in the backward direction (in bytes)
13	std_bpktl	The standard deviation from the mean of the packets sent in the backward direction (in bytes)
14	min_fiat	The minimum amount of time between two packets sent in the forward direction (in microseconds)
15	mean_fiat	The mean amount of time between two packets sent in the forward direction (in microseconds)
16	max_fiat	The maximum amount of time between two packets sent in the forward direction (in microseconds)
17	std_fiat	The standard deviation from the mean amount of time between two packets sent in the forward direction (in microseconds)
18	min_biat	The minimum amount of time between two packets sent in the backward direction (in microseconds)
19	mean_biat	The mean amount of time between two packets sent in the backward direction (in microseconds)
20	max_biat	The maximum amount of time between two packets sent in the backward direction (in microseconds)
21	std_biat	The standard deviation from the mean amount of time between two packets sent in the backward direction (in microseconds)
22	duration	The duration of the flow (in microseconds)
23	min_active	The minimum amount of time that the flow was active before going idle (in microseconds)
24	mean_active	The mean amount of time that the flow was active before going idle (in microseconds)
25	max_active	The maximum amount of time that the flow was active before going idle (in microseconds)
26	std_active	The standard deviation from the mean amount of time that the flow was active before going idle (in microseconds)
27	min_idle	The minimum time a flow was idle before becoming active (in microseconds)
28	mean_idle	The mean time a flow was idle before becoming active (in microseconds)
29	max_idle	The maximum time a flow was idle before becoming active (in microseconds)
30	std_idle	The standard deviation from the mean time a flow was idle before becoming active (in microseconds)
31	sflow_fpackets	The average number of packets in a sub flow in the forward direction
32	sflow_fbytes	The average number of bytes in a sub flow in the forward direction
33	sflow_bpackets	The average number of packets in a sub flow in the backward direction
34	sflow_bbytes	The average number of packets in a sub flow in the backward direction
35	fpsh_cnt	The number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
36	bpsh_cnt	The number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
37	furg_cnt	The number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
38	burg_cnt	The number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
39	total_fhlen	The total bytes used for headers in the forward direction.
40	total_bhlen	The total bytes used for headers in the backward direction.

specify which data is HTTPS and which data is Tor, to accomplish that, a labeling process is required by specifying the label attribute on each data instance in the ARFF file. However, this data is known as ground-truth. Building up ground-truth is very important and critical phases of any traffic classification method since the entire classification process relies completely on the accuracy of this labeling. Thus, accuracy is important by labeling data instances based on their types to ensure the minimum false positives and false negatives. Also, because traffic has been completely separated based on CVM(i), validation has been conducted to ensure only traffic generated by each CVM is corresponded to that CVM, and no other traffic noise mixed up with the intended traffic to be used.

D. Building Ml Classification Model

Supervised ML is employed in order to create the training dataset. In Supervised learning ML; the algorithm takes a known data called training dataset to make some predictions. The method attempts to discover the relationship between input attributes and target attributes, the output relationship discovered represents a structure called "model" see Fig. 6.

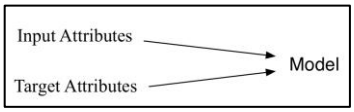

Fig. 6. Generating Model in ML using supervised learning

Weka is an open source project that contains different tools for data pre-processing, regression, classification, clustering, association rules, and visualization, and can be used directly by providing a dataset or from a java code, as in Fig. 7.

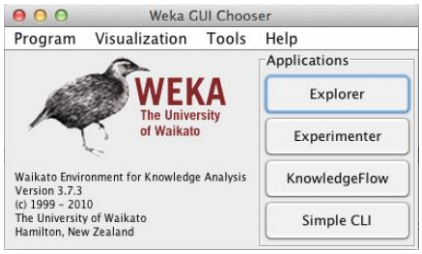

Fig. 7. Weka GUI in OSX

In order to apply ML algorithms and build the data model, given the different traffic data training sets, Weka is chosen for this exercise. Weka GUI or direct command line interface can be used to accomplish this, Fig. 8 below presents the use of Weka as a simple command line interface to generate a data model.

$$java\ -classpath\ weka.jar\ weka.CLASSIFIERS.TYPE \\ -t\ training.rff\ -d\ TYPE.model$$

Fig. 8. Creating data model using Weka CLI

V. Evaluation Techniques

In this experiment, in order to fingerprint websites over Tor, a few ML methods were used. The experiment was repeated multiple times using Weka, each time using different set of training and test cases (changing the number of packets used to create the case). However, to obtain a simulated test performance, the testing data used in the evaluation are the same as the training data set but with 10 cross-validation using Weka.

Cross-validation means that part of the data will be reserved for testing while the rest will be used for training. In other words, the data is partitioned into 10 parts (folds), one part for testing and the remaining 9 parts for training. Further, different set of attributes (features) used for classification, and a deep investigation has been performed in order to find relevant attributes and building minimal rule sets for classifying Tor traffic (finding the minimal rule set is proved to be an NP-hard problem [25]) and different classification test cross-validation option to achieve higher accuracy with less FPR and FNR. Fig. 9 diagram shows the steps of classification method in general.

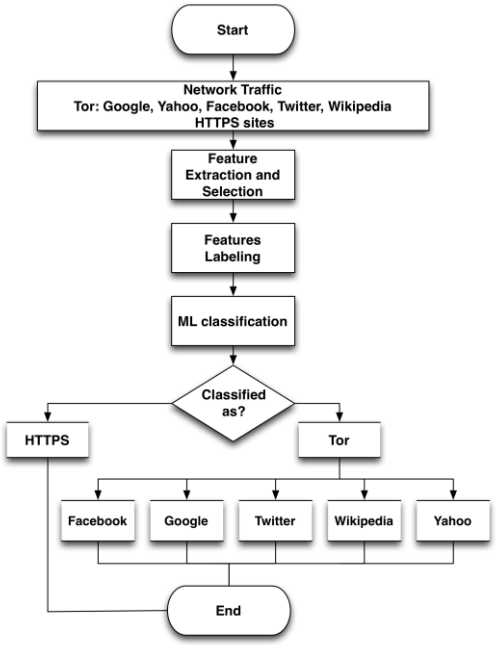

Fig. 9. Detection Diagram using ML

The sites used fingerprinting evaluation is the top 5 websites on Alexa, the sites are listed with a localization domain to avoid Tor redirection into the local IP location. Table VI ists the top websites that have been chosen in the evaluation process.

Table 6. List of websites used in the fingerprinting process

#	Site
Google	https://www.google.de
Facebook	https://www.facebook.com
Yahoo	https://se.yahoo.com
Twitter	https://www.twitter.com
Wikipedia	https://www.wikipedia.org/

A. Classification Methods Employed

The focus of this research is to employ ML methods in classifying Tor encrypted flows, classifying is considered based on the number of packets necessary to correctly classify those flows and the number of feature sets used. According to researcher knowledge, there is no any other research that has exclusively worked to fingerprint and classify Tor traffic amongst other HTTPS traffic. Below is a description of each ML methods used in the experiment. Each method is used to classify data collected from Tor amongst HTTPS data, the variation in the flow characteristics can be understood by each ML algorithm in order to provide the classification accuracy.

The goal is to achieve high accuracy with low FP in order for the methodology to successfully fingerprint Tor sites on a local network environment.

1) Classification Using Statistical Model

Naïve Bayes is used in the evaluation methodology in order classify Tor and HTTPS traffic. Naïve Bayes is a classification algorithm that relies on Bayes' rule of conditional probability [26]. Naïve Bayes ML technique forms a statistical model of data that is given in the training phase. The algorithm relates each feature to the probability that feature will result in a particular outcome based on the entire training set. To preform testing, the probability of each possible outcome is calculated based on the features each test instance has. Naïve Bayes gets its name because it makes the (naive) assumption that each feature is independent, and uses Bayes rule of conditional probability.

2) Classification Using Decision Trees

The C4.5 is a decision tree classifier, which is built by repeatedly splitting the training set on the feature (attribute), which "best" splits, the data. Thus, the consideration is to use it in order to classify Tor and HTTPS and provide high accuracy results. There are multiple methods for deciding which feature is best, but C4.5 uses a measure of information entropy. The exact criterion for splitting the training set is the normalized information gain, which is the difference in entropy caused by choosing a specific attribute for splitting the data. The attribute that has the highest normalized information gain is ultimately selected to be the one on which the training set is split. The resulting model of C4.5 is, in effect, a series of IF/THEN statements, which do not necessarily employ all attributes. Given this structure, there may be multiple paths for the same outcome class.

3) Random Forest

Random forest or (RF) is a ML algorithm that evolved from decision trees, and used in this classification to ensure the results are aligned with what is achieved by both Naïve Bayes and C4.5 and because of classification strength of the algorithm. RF consists of many decision trees and supports two ML algorithms bagging and random selection. In bagging, short for 'bootstrap aggregating', and one of the first ensemble methods, ensemble methods are based on the idea that by combining multiple weaker learners, a stronger learner is created [27], the prediction is made based on the majority of trees votes by training each tree on a bootstrap sample of training sample data. Random feature selection conducts a simple search to find the best split in each node while growing a tree over a random subset of features.

4) Support Vector Machine

The support vector machine (SVM) that is first pioneered by Vapnik and Chervonenkis [28] and is the state-of-art supervised ML algorithm for the binary classification problem. SVM is heavily used for data mining and is very well known by its high performance in terms of the classification accuracy, thus, it has been considered in this research to make sure the results that are achieved are not biased to specific ML algorithm and that this type of ML classification is also capable to classify Tor amongst HTTPS traffic. In SVM, Given a set of objects that falls into two categories (training data), the problem is how to classify a new point (test data) into one of the aforementioned categories. SVM solves this issue by calculating the line in which the data can be separated into two categories, training and test data [29].

The key idea of SVM is the interpretation of instances as vectors, in this research classification problem, the instances are the data generated through site retrieval, which represented as vectors. However, based on the training data provided, the SVM classifier tries to fit a hyperplane into the vector multidimensional space which represents the instances in order to create a separation between the instances that are belong to a different class. The accumulated distance between the fitted plane and the support vectors (instances) has to be as high as possible where it needs to maximizes the gap between the two classes. However, sometimes the vectors are not linearly separable and require complex decision planes for optimal separation of the categories similar to Fig. 10. Which SVM can solve transforming the vector space into a higher dimensional space by the so-called kernel trick, in the higher dimension; the hyperplane can be fitted again.

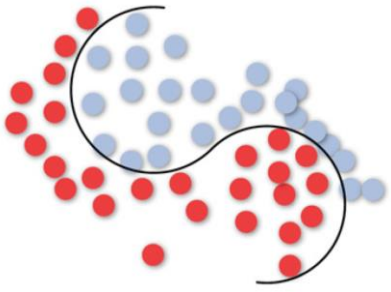

Fig. 10. Nonlinear SVM separator

VI. Results and Discussions

This section presents the evaluation results of the classification experiment for fingerprinting Tor encrypted traffic in the offline traffic traces, which has been discussed previously. HTTPS and Tor-SSL traffic have been used in order to create the training dataset. The main goal of this research is to evaluate the possibility of

providing a high detection rate for Tor traffic amongst HTTPS traffic in order to fingerprint most monitored websites on Alexa from a local observer sitting on the network.

In this research's traffic classification for Tor; two factors are typically considered in order to quantify the performance of the classifier: Detection Rate (DR) and False Positive Rate (FPR). In this case DR or accuracy will reflect the number of Tor-SSL flows correctly classified whereas FPR will reflect the number of HTTPS flows incorrectly classified as Tor-SSL. Naturally, a high DR rate and a low FPR would be the desired outcomes for us [30]. DR and FPR are calculated based on the following equations:

$$\mathrm{DR} = 1 - \frac{\#\mathrm{FNClassifications}}{\mathrm{TotalNumberTorClassifications}} \quad (10)$$

$$\mathrm{FPR} = \frac{\#\mathrm{FPClassifications}}{\mathrm{TotalNumberNonTorClassifications}} \quad (11)$$

In equation (10), FN represents False Negative, which means Tor-SSL traffic classified as HTTPS traffic. Likewise, in equation (11), FPR represents false positive rate, which means HTTPS traffic classified as Tor-SSL traffic. Since the main goal is to achieve a high DR rate and a low FP rate results, the experiment has been evaluated by four ML algorithms using Weka [31]. However, In order to evaluate the accuracy/errors of using ML. The experiment has been run with 10-cross validation set option in Weka, cross validation is a necessary step in model construction, it assesses how the results of a statistical analysis will generalize to an independent data set and provides estimation on how this model will perform in practice.

A. Classifiers Results

This research goal is to achieve High true positive rate sometimes known as DR and less FPR. In order to attain that, a feature selection exercise has been performed, feature selection would eliminate features determined to be of a little use in classifying and reducing the computations needed, feature selection used by tuning the features calculated from the training packets of the flow, a high accuracy have been achieved using different features set, this also improved the runtime of the ML algorithms that require intensive mathematical calculations, data has been generated on a local network environment by following the best approach described in Ian and Tao method [14] for Tor dataset generation. In the experiment, the fingerprinting has been performed on the top 5 monitored sites on Alexa, the sites are Google, Facebook, Yahoo, Twitter, and Wikipedia, those sites running various types of content and serving almost more than 100 million users every day. Breaki

ng Tor anonymity meaning a direct identification of those traffic instances within the network traffic. The researcher has performed the fingerprinting using ML classification technique; four ML algorithms have been used to classify the traffic and all has shown very close results. The accuracy, time training, and runtime including some analysis are described below.

1) Naïve Bayes

Naïve Bayes is a classification algorithm that relies on Bayes' rule of conditional probability [26]. In the experiment, Naïve Bayes in Weka is used to classify Tor instances with 10-fold cross-validation test mode, by using 40 features, Naïve Bayes was able to achieve a high TP Rate 99.60% and FP Rate 0.004% and 99.69% accuracy. Table VII is breakdown of the detailed accuracy using Naïve Bayes for each monitored site with the weighted average details; the weighted average is computed by weighting the measure of class (TP Rate, FP Rate, Precision, Recall, F-Measure, ROC Area) by the proportion of instances in that class. Fig. 11 also provides an overall distribution in a chart presentation.

Table 7. Breakdown of Naïve Bayes classification for top monitored sites

Class	*TP Rate*	*FP Rate*	*Precision*	*Recall*	*F-Measure*	*ROC Area*
Tor Google	0.991	0.002	0.99	0.991	0.99	0.999
HTTPS	0.998	0.009	0.998	0.998	0.998	0.994
Weighted Avg	0.997	0.008	0.997	0.997	0.997	0.994
Tor Facebook	0.994	0.002	0.988	0.994	0.991	0.997
HTTPS	0.998	0.006	0.999	0.998	0.999	0.997
Weighted Avg	0.998	0.005	0.998	0.998	0.998	0.997
Tor Yahoo	0.998	0.001	0.995	0.998	0.996	0.999
HTTPS	0.999	0.002	0.999	0.999	0.999	0.998
Weighted Avg	0.999	0.002	0.999	0.999	0.999	0.998
Tor Twitter	0.992	0	0.999	0.992	0.995	0.999
HTTPS	1	0.008	0.998	1	0.999	0.996
Weighted Avg	0.998	0.007	0.998	0.998	0.998	0.996
Tor Wikipedia	0.993	0.007	0.954	0.993	0.973	0.998
HTTPS	0.993	0.007	0.999	0.993	0.996	0.996
Weighted Avg	0.993	0.007	0.993	0.993	0.993	0.997

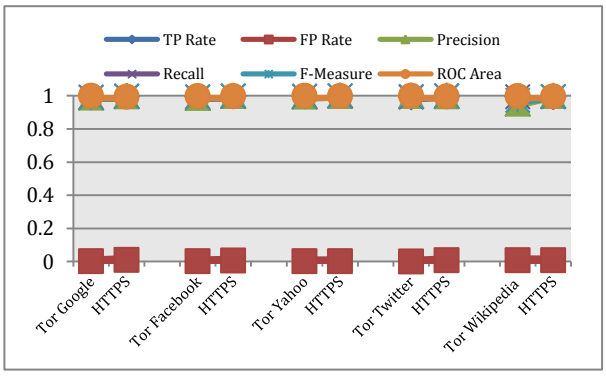

Fig. 11. Distribution of Na ïve Bayes classification for each monitored site

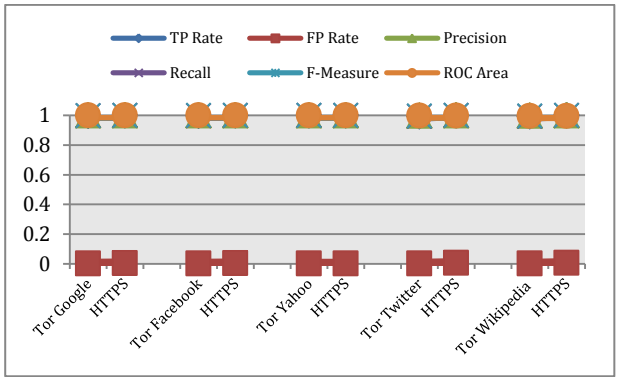

Fig. 12. Distribution of Na ïve Bayes classification for each monitored site.

2) *C4.5*

C4.5 is a decision tree classifier, one of the amazing features about C4.5 is the determination of how deeply to grow a decision tree to avoid overfitting and choosing an appropriate attribute selection measures. Table VIII shows the result of using C4.5 with 10-fold cross-validation test mode, C4.5 is known as J48 in Weka. Fig 12 shows the overall distribution in chart representation, C4.5 achieves higher accuracy compared to Na ïve Bayes with 99.92% and 99.85% TP Rate, 0.002% FP Rate.

Table 8. Breakdown of C4.5 classification for monitored sites

Class	*TP Rate*	*FP Rate*	*Precision*	*Recall*	*F-Measure*	*ROC Area*
Tor Google	0.997	0	0.999	0.997	0.998	0.999
HTTPS	1	0.003	1	1	1	0.999
Weighted Avg	0.999	0.002	0.999	0.999	0.999	0.999
Tor Facebook	0.998	0	0.999	0.998	0.999	0.999
HTTPS	1	0.002	1	1	1	0.999
Weighted Avg	1	0.002	1	1	1	0.999
Tor Yahoo	0.999	0	1	0.999	1	0.999
HTTPS	1	0.001	1	1	1	0.999
Weighted Avg	1	0.001	1	1	1	0.999
Tor Twitter	0.994	0.001	0.997	0.994	0.996	0.997
HTTPS	0.999	0.006	0.999	0.999	0.999	0.997
Weighted Avg	0.999	0.005	0.999	0.999	0.999	0.997
Tor Wikipedia	0.994	0	0.998	0.994	0.996	0.996
HTTPS	1	0.006	0.999	1	0.999	0.996
Weighted Avg	0.999	0.005	0.999	0.999	0.999	0.996

3) *Random Forest*

This algorithm evolved from decision trees and supports bagging and random selection, random forest performs much faster than boosting and bagging. The results for the classification shows that Random forest achieved the higher TP Rate results compared to Na ïve Bayes and C4.5 with 99.92% accuracy and 99.86% TP Rate, 0.002% FP Rate as described in Table IX and Fig. 13, the algorithm is run using 10-fold cross-validation test mode.

Table 9. Breakdown of Random Forest classification for top monitored sites

0	*TP Rate*	*FP Rate*	*Precision*	*Recall*	*F-Measure*	*ROC Area*
Tor Google	0.996	0	0.999	0.996	0.998	1
HTTPS	1	0.004	0.999	1	1	1
Weighted Avg	0.999	0.003	0.999	0.999	0.999	1
Tor Facebook	0.998	0	1	0.998	0.999	1
HTTPS	1	0.002	1	1	1	1
Weighted Avg	1	0.002	1	1	1	1
Tor Yahoo	0.999	0	1	0.999	1	1
HTTPS	1	0.001	1	1	1	1
Weighted Avg	1	0.001	1	1	1	1
Tor Twitter	0.995	0.001	0.997	0.995	0.996	0.999
HTTPS	0.999	0.005	0.999	0.999	0.999	0.999
Weighted Avg	0.999	0.004	0.999	0.999	0.999	0.999
Tor Wikipedia	0.995	0	0.997	0.995	0.996	0.999
HTTPS	1	0.005	0.999	1	0.999	0.999
Weighted Avg	0.999	0.004	0.999	0.999	0.999	0.999

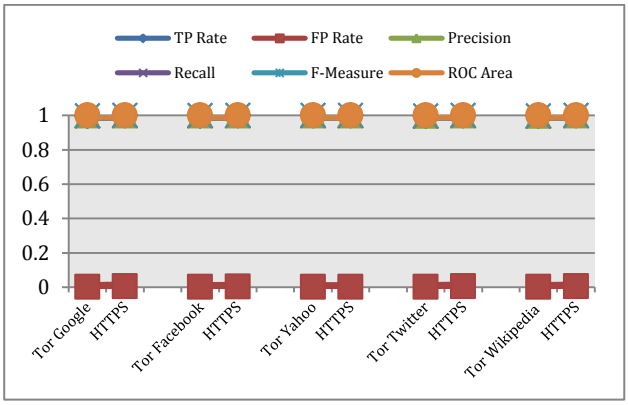

Fig. 13. Distribution of Random Forest classification for each monitored site

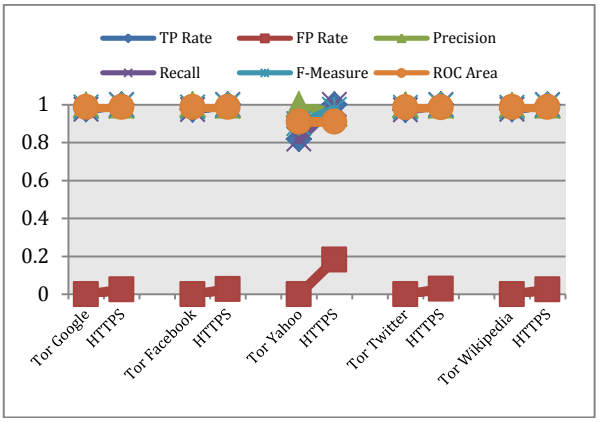

Fig. 14. Distribution of SVM classification for each monitored site.

4) *SVM*

SVM is the state-of-the-art supervised ML method, most of the previous studies on Tor fingerprinting used SVM as classifier [14]. Thus, the researcher has considered SVM in order to ensure the results achieve better accuracy confirming the improvement of the method considered in this research regardless of the methodology used for Tor fingerprinting. SVM achieved an accuracy of 99.04% and 97.72% TP Rate, 0.034% FP Rate with 10-fold cross-validation test mode. Table X and Fig. 14 are the complete detailed results.

Table 10. Breakdown of SVM classification for top monitored sites

Class	*TP Rate*	*FP Rate*	*Precision*	*Recall*	*F-Measure*	*ROC Area*
Tor Google	0.975	0	0.999	0.975	0.987	0.987
HTTPS	1	0.025	0.996	1	0.998	0.987
Weighted Avg	0.996	0.022	0.996	0.996	0.996	0.987
Tor Facebook	0.974	0	1	0.974	0.987	0.987
HTTPS	1	0.026	0.996	1	0.998	0.987
Weighted Avg	0.997	0.023	0.997	0.997	0.997	0.987
Tor Yahoo	0.818	0	1	0.818	0.9	0.909
HTTPS	1	0.182	0.953	1	0.976	0.909
Weighted Avg	0.961	0.143	0.963	0.961	0.96	0.909
Tor Twitter	0.972	0	0.999	0.972	0.985	0.986
HTTPS	1	0.028	0.995	1	0.997	0.986
Weighted Avg	0.995	0.024	0.995	0.995	0.995	0.986
Tor Wikipedia	0.975	0	0.998	0.975	0.986	0.987
HTTPS	1	0.025	0.996	1	0.998	0.987
Weighted Avg	0.996	0.022	0.996	0.996	0.996	0.987

Basically the four algorithms Naïve Bayes, C4.5, Random forests, and SVM achieved almost very similar results as shown in Fig. 15 for all top monitored sites with less accuracy achieved for both Twitter and Wikipedia considering the dynamic content in both sites. Also less TP results achieved for Yahoo classification using SVM.

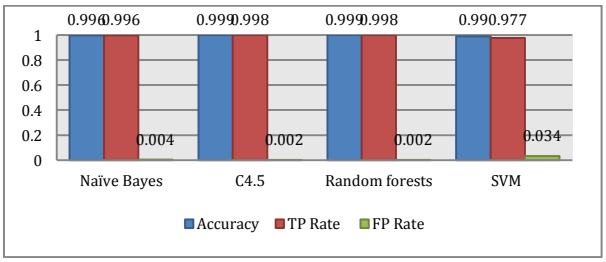

Fig. 15. Results comparison between ML algorithms

B. *Comparison*

In order to compare this research results with previous achieved results on Tor fingerprinting, the researcher needs to perform the improved methodology on the same data used in previous researches and compare results. However, because Tor literature covers a wide verity of techniques with many different goals, and no two techniques can be directly compared, as the data used for analysis is not publicly disclosed [32]. The researcher used some parameters for data generation technique (Tao Wang I. G., 2013) which previously known to achieve higher accuracy results ignoring the removal of SENDMEs as it did not affect the results that much and then use this dataset with the improved methodology to present the new results.

There are a couple of few researches that have been known to achieve high fingerprinting results on some monitored sites. The recent research by [14] had achieved an accuracy of 91% using SVM. However, the methodology used in that research is different on how fingerprinting technique works, the accuracy achieved in this research using SVM is 99.04%, which gives an improvement of +8.04%. Fig. 6 shows a graph comparison between both results.

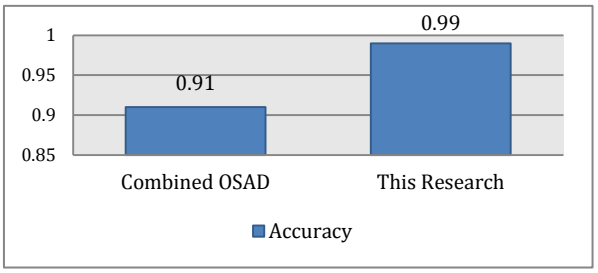

Fig. 16. Combined OSAD accuracy versus this research accuracy.

Considering the technique used in order to identify Tor traffic, this research by [33] were able to identify Tor traffic using ML and they proofed that simulated Tor network can be distinguished from regular encrypted traffic, suggesting that real world Tor users may be vulnerable to the same analysis. Barker et al [33] were able to detect Tor over HTTP and Tor over HTTPS. Further, he was able to achieve a result of 90% using different ML algorithms. Basically, their evaluation is based on the size of individual packets in a stream as feature for traffic classification. However, this research is considered a similar approach to distinguish Tor traffic from HTTPS in order to achieve websites fingerprinting over Tor. Yet, employing different improved techniques and different feature set for the evaluation, this research results gave an improvement, in Random forest of +2.1% and for C4.5 of +2.8%. Fig. 17 and Fig. 18 show a comparison between both results considering the mutual ML algorithms used Random Forest and C4.5.

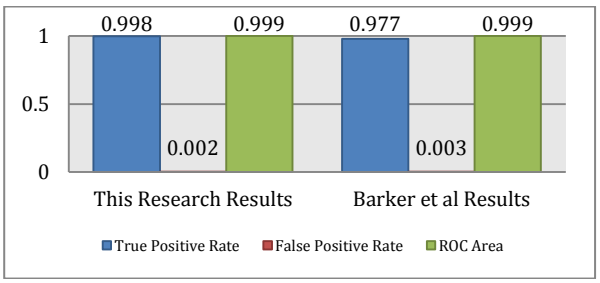

Fig. 17. Results comparison between John and This research accuracy using Random Forest ML algorithm

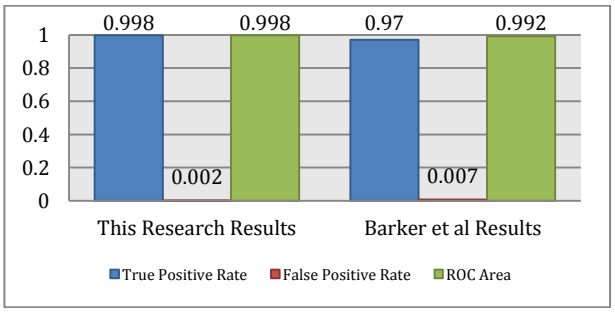

Fig. 18. Results comparison between John and This research accuracy using C4.5 ML algorithm

C. Discussion

In this paper, the researcher has demonstrated a website fingerprinting attack against the most widely known anonymity project Tor. Tor is very hard to detect by measuring one parameter, in our methodology, the researcher presented an improved technique in order to fingerprinting websites over Tor network, the method combines various improvements in order to achieve higher results amongst previous researches on Tor fingerprinting. The results have shown that all ML algorithms employed achieved very similar results, almost 99% for all top sites on Alexa, meaning the accuracy achieved is not biased to a specific ML algorithm and that the variation is in the existence of in the characteristics of Tor traffic amongst HTTPS traffic.

According to Tor project, the assumption for Tor is that data over Tor and HTTPs traffic should look alike, preventing the local observer from distinguishing both traffic traces in Tor, and thus preserving privacy. However, this research results refute this assumption by noticing that Tor and HTTPS traffic have different flow characteristics, which proofed by showing high accuracy on distinguishing Tor and HTTPS traffic. Yet, this implies that Tor protections are not enough to make both traffic traces look alike in order to preserve users' privacy and this indicates that the current protections in Tor implementation breaks the anonymity that Tor promised. The variations in flow characteristics can be shown in Fig. 19 the figure shows a sample traffic that is taken from ARFF file represents Google Tor traffic in blue and HTTPS traffic in red. As described in Table V , the features represent different network traffic flows. Also Fig. 20 shows a plot matrix in Weka for the current dataset, Tor Google traffic in blue and HTTPS in red, the plot matrix shows the distribution for each class feature amongst the other class features in a matrix distribution form. From the chart, its obvious the variation between each class of traffic instances for the current sample provided in Fig. 20 However, in the evaluation, 40 features were used which gave a high classification accuracy for both Tor and HTTPS traffic.

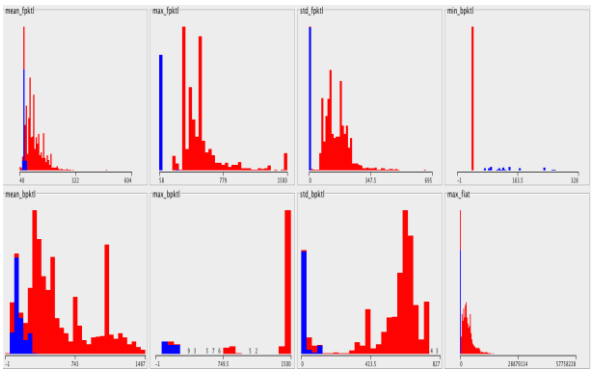

Fig. 19. Variation in flow characteristics of sample Google's ARFF file

D. Conclusions And Future Work

This research presents that Tor can be classified amongst HTTPS encrypted traffic. Tor is the low-latency anonymity tool and one of the prevalent used open source anonymity tools for anonymizing TCP traffic on the Internet. Tor has implemented different defenses techniques in order to prevent automated identification of Tor traffic such as TLS encryption, padding, and packet

relaying. However, as proofed in this research, Tor does not appear to appropriately succeed in blurring the network packets features, which makes it possible for a local observer to identify Tor traffic in the network and fingerprint most top sites on Alexa. Different techniques have been used in order to classify Tor, similar technique in previous researches is used to generate the traffic and dataset model, Netmate is used for features dump and Weka is used to build the dataset model, several ML algorithms have been employed to identify Tor traffic, results gave an improvement amongst previous results by achieving an accuracy of 99.64% and 0.01% FP. However, the researcher believes that its hard to compare this research results with previous researches as Tor literature covers a wide variety of techniques with many different goals, and no two techniques can be directly compared as the data used for analysis is not publicly disclosed.

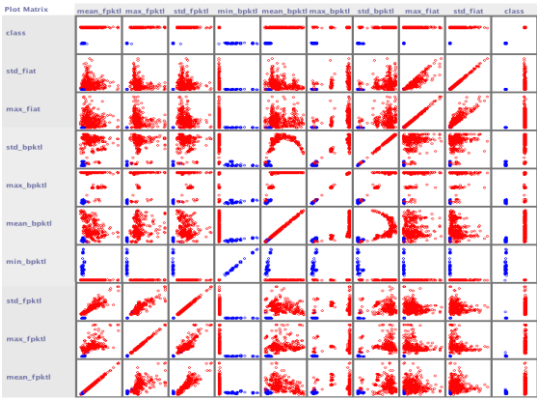

Fig. 20. Plot Matrix for a sample of Google's ARFF file

E. Recommendations for Future Work

The researcher believes that this research experiment was based on a small set of simulated data, and thus, it is not necessarily that it covers all possible real world conditions including open world experiments. The noise and variability present in the real Tor network may make this classification technique inaccurate. As future recommendation, it's important to involve different types of noise in the dataset to mimic the real open world experience The researcher believes it's important to study the ability to classify Tor on a global scope like ISP or even more with some fine-tuning to the parameters used in the experiments. Also due to high computation costs of SVM, it's important to use a parallel computing cluster to perform the experiment. Further, increasing the scope of fingerprinting to include more sites in the experiment and study the variation in the accuracy for each. Finally, as future recommendation for Tor protocol, the researcher advices that developers should develop more defenses in order to make it harder for local observer to classify Tor amongst HTTPS traffic and thus pertain anonymity and privacy for Tor users.

REFERENCES

[1] Inc Tor Project. (2012, July) torproject. [Online]. https://metrics.torproject.org

[2] J. R. Vacca, Computer and information security handbook.: Newnes, 2012.

[3] B Schneier, Schneier on security.: John Wiley & Sons, 2009.

[4] M., Adair, S., Hartstein, B., & Richard, M Ligh, Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code.: Wiley Publishing, 2010.

[5] B., Erdin, E., Güneş, M. H., Bebis, G., & Shipley, T. Li, An Analysis of Anonymizer Technology Usage. Berlin: Springer, 2011.

[6] X., Zhang, Y., & Niu, X. Bai, "Traffic identification of tor and web-mix," in In Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference, 2008, pp. 548-551.

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

[8] P. Loshin, Practical Anonymity: Hiding in Plain Sight Online.: Newnes, 2013.

[9] Edward M. Schwalb, iTV handbook: technologies & standards.: Prentice Hall, 2003.

[10] Manuel Mogollon, Cryptography and Security Services: Mechanisms and applications.: CyberTech Publishing, 2007.

[11] M., Klonowski, M., & Kutyłowski, M. Gomułkiewicz, "Onions based on universal re-encryption–anonymous communication immune against repetitive attack," in In Information Security Applications, Berlin , 2005, pp. 400-410.

[12] E., Shin, J., & Yu, J. Chan-Tin, "Revisiting Circuit Clogging Attacks on Tor," In Availability, Reliability and Security (ARES), 2013 Eighth International Conference, pp. 131-140, 2013.

[13] Nick Mathewson Roger Dingledine. (2004) torproject. [Online].https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=tor-spec.txt

[14] T., & Goldberg, I. Wang, "Improved website fingerprinting on tor," in In Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society, New York, 2013, pp. 201-212.

[15] Z., Luo, J., Yu, W., Fu, X., Xuan, D., & Jia, W. Ling, "A new cell-counting-based attack against Tor," IEEE/ACM Transactions on Networking (TON), vol. 20(4), pp. 1245-1261, 2012.

[16] S., Nguyen, T., & Armitage, G. Zander, "Automated traffic classification and application identification using machine learning," in In Local Computer Networks, 2005. 30th Anniversary, 2005, pp. 250-257.

[17] Selenium. (2004) Selenium. [Online]. http://docs.seleniumhq.org/

[18] Margaret Rouse. (2011) search server virtualization. [Online].http://searchservervirtualization.techtarget.com/definition/virtual-machine

[19] (1987) Tcpdump. [Online]. http://www.tcpdump.org/

[20] Alexa. (1996) Alexa. [Online]. http://www.alexa.com/

[21] The Fraunhofer Institute for Open Communication Systems FOKUS. (2010) ip-measurement. [Online]. http://www.ip-measurement.org/tools/netmate.

[22] C., & Zincir-Heywood, A. N. McCarthy, "An investigation on identifying SSL traffic," In Computational Intelligence for Security and Defense Applications (CISDA), pp. 115 - 122, 2011.

[23] University of Waikato. (2008) Waikato. [Online]. http://www.cs.waikato.ac.nz/ml/weka/arff.html

[24] O., & Rokach, L. Maimon, "Introduction to supervised methods," In Data Mining and Knowledge Discovery Handbook, pp. 149-164, 2005. [Online]. http://www.ise.bgu.ac.il/faculty/liorr/hbchap8.pdf
[25] J. Wroblewski, "Finding minimal reducts using genetic algorithms," in In Proccedings of the second annual join conference on infromation science, 1995, pp. 186-189.
[26] I. H., Gori, M., & Numerico, T. Witten, Web dragons: Inside the myths of search engine technology.: Elsevier, 2010.
[27] B. Lantz, Machine Learning with R.: Packt Publishing Ltd, 2013.
[28] V. N., & Chervonenkis, A. J. Vapnik. (1974) Theory of pattern recognition.
[29] V. Agneeswaran, Big Data Analytics Beyond Hadoop: Real-Time Applications with Storm, Spark, and More Hadoop Alternatives.: Pearson Education, 2014.
[30] R., & Zincir-Heywood, A. N. Alshammari, "Machine learning based encrypted traffic classification: identifying ssh and skype," in In Computational Intelligence for Security and Defense Applications, 2009, pp. 1-8.
[31] University of Waikato. (2013) [Online]. http://www.cs.waikato.ac.nz/ml/weka/
[32] N., Zander, S., & Armitage, G. Williams, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," ACM SIGCOMM Computer Communication Review, pp. 5–16, 2006.
[33] J., Hannay, P., & Szewczyk, P. Barker, "Using traffic analysis to identify The Second Generation Onion Router," in In Embedded and Ubiquitous Computing (EUC), 2011 IFIP 9th International Conference, 2011, pp. 72-78.

Authors' profiles

Mr. Almubayed is a security researcher was born in 1985 and received his B.S degree from Al-Balqa applied university (BAU) in 2008. Recently he has completed his MS degree in information security and digital crimes from Princess Sumaya University for Technology (PSUT), Amman in 2014.

Mr. Almubayed worked as a software developer with various software companies in Jordan. In 2009, he joined Maktoob, which later acquired by Yahoo Inc. He is currently based in Sunnyvale, California, and works with Yahoo inc!, as a security engineer. Mr. Almubayed has conducted researches in various areas, including web defensive tools, employing machine learning for traffic classifications, and he has more than 5 years of experience in the fields of information security, ethical hacking, reverse engineering, risk management, and computer programming.

Dr. Hadi received the B.S. degree in computer science from Philadelphia University, Jordan, in 2002 and the M.Sc. and Ph.D. degree in computer information system from University of Banking and Financial Sciences, College of Information Technology, Jordan, in 2004 and 2010, respectively.

He's a Senior Level Information Security Officer with 14+ years of professional experience working for different high-reputed companies. Since 2011 he's been teaching different computer security, digital forensics, and networking courses for both graduates and undergraduates. He's also an author, speaker, and freelance instructor. His research interests include digital forensics, operating systems internals, malware forensic analysis, and network security.

Prof. Atoum is currently the Dean of The King Hussein School of Computing Sciences at Princess Sumaya University for Technology (PSUT). He had received his B.S. degree in Computer Science from Yarmouk university-Jordan in 1984. He had received his Master degree in Computer Science from University of Texas at Arlington-USA in 1987. He had received his PhD in Computer Science from University of Houston-USA in 1993. He had worked as an assistant professor at Yarmouk University from 1993 to 1995. He had been appointed as the Computer Science department Chairman at PSUT. He has supervised or co-supervised several students on their Ph.D. dissertations and several M.S. theses and has supervised numerous undergraduate graduation projects. Finally, he have been involved in several committees for degree plans, proposed and developed the Master program in Information System Security and Digital Criminology at PSUT.

How to cite this paper: Alaeddin Almubayed, Ali Hadi, Jalal Atoum,"A Model for Detecting Tor Encrypted Traffic using Supervised Machine Learning", IJCNIS, vol.7, no.7, pp.10-23, 2015.DOI: 10.5815/ijcnis.2015.07.02

