Detection of Tor traffic using deep learning

Debmalya Sarkar
Department of Information and Communication Technology
Manipal Institute of Technology, MAHE, Manipal-576104, India.
debmalya97@gmail.com

Vinod P.
Department of Computer Science and Engineering,
SCMS School of Engineering and Technology, Kerala, India.
vinodp@scmsgroup.org

Suleiman Y. Yerima
Cyber Technology Institute,
Faculty of Computing, Engineering & Media, De Montfort University,
Leicester, United Kingdom.
syerima@dmu.ac.uk

***Abstract*— Tor, originally known as The Onion Router, is a free software that allows users to communicate anonymously on the Internet. This makes Tor attractive to cyber criminals, and the anonymity provided can be misused by hackers to enable remote control of victim systems. Indeed, a large volume of Tor traffic is used for malicious purposes such as fast port scans, hacking attempts, ex-filtration of stolen credentials, etc. This makes Tor traffic detection an important component of intrusion detection and prevention systems. Hence, in this paper we present a deep neural network (DNN) based system for the detection and classification of encrypted Tor traffic. The system achieved 99.89% accuracy in the classification of Tor and non-Tor traffic on the UNB-CIC Tor network dataset. Experiments conducted for classifying Tor traffic types demonstrated an accuracy of 95.6%, which is 6.2% higher than previous work on the same dataset. Additionally, the robustness of the proposed DNN classifier is evaluated using adversarial samples generated from a Generative Adversarial Network (GAN). We observed that 100% of the adversarial examples were unidentified by the DNN classifiers. Further retraining of the DNN classifiers with adversarial examples eventually improved their robustness against the adversarial attack.**

I. Introduction

In the present era of fast-paced Internet omnipresence and connectivity, any information needed by end-users is readily available in mere seconds. This ease of access to the Internet raises concerns regarding the release of sensitive data and individual browsing behaviour without the permission of users. Nowadays, it is quite common for organizations to closely monitor inbound/outbound network traffic to detect patterns indicative of anomalous behaviour. Thus, traffic analysis and classification has emerged as an active research area over the last few years. Traffic classification has found widespread application in preventing cyber-attacks [1], optimizing the underlying network [2], relocating network resources for prioritized users, understanding types of transit traffic [3], diagnostics monitoring, network management, etc. Traditional traffic classification methods use knowledge of packet headers for determining suspicious behaviour. This approach based on signature matching is inefficient, as it is incapable of identifying new instances. Hence, there is a need to develop more efficient and more effective methods of performing traffic analysis.

Currently, organizations are keen on adopting security solutions capable of preserving the privacy of user data. Commonly adopted methods include the use of dynamic port numbers and data encryption before leaving the secured network. Tor browser [4] is a popular application used to provide anonymity to its users. Tor permits users to circumvent internet censorship by routing data over relay servers acting as proxies. Initially, a user browses the web by anonymously sending a request to the directory of Tor servers. After establishing a connection, data is passed to the first server also known as an entry guard. The request is then forwarded to a one-hop server that uses session keys of the previous server. Eventually, the last hop relays data to the destination. Anonymity is thus ensured by hiding the IP addresses, incorporating three routers, and session keys.

While Tor allows anonymous communication over the Internet, it is also prone to abuse by activists, hackers and cybercriminals. Tor is increasingly used for illegal activities such as sharing copyrighted materials, selling weapons, sharing pornography etc. Furthermore, attackers may employ the Tor service to run command and control (C&C) infrastructure to communicate with botnets.

Intelligent systems employing machine learning approaches have been developed to perform accurate classification of network traffic [5], [6]. Such solutions balance the trade-off between computational complexity and classification accuracy quite successfully, with a majority of frameworks using shallow architectures [7]. With the explosion of bulk internet traffic, the number of packets required to be analyzed is continually increasing. Thus, it is infeasible for a network administrator to analyze all of them. Consequently, the performance of traditional classification algorithms is far from satisfactory when deployed on large scale training samples and features.

The last few years have witnessed the evolution of deep learning architectures [8], [9] especially in domains such as computer vision [10] and text analytics [11]. These architectures have multiple layers consisting of non-linear processing units, and can appropriately identify relevant attributes from a large set of training examples to infer meaningful patterns. Hence, these networks have been successfully leveraged for traffic classification [12], [13] phishing detection [14], [15] and botnet detection [16].

In this paper, we present an effective method to differentiate between *Tor* and *non-Tor* traffic using deep neural networks. Additionally, we show that it is possible to detect the traffic type from the encrypted flows, thus allowing the

network owner (for example an ISP) to block specific content transmitted over the Tor network. In order to evaluate the robustness of the developed deep neural network models, we implemented a traffic generator using Generative Adversarial Network[17]. In particular, we use the trained generator to create malicious traffic with a statistical distribution identical to legitimate ones to fool the previously trained machine learning models. The trained models failed to detect any of the adversarial examples despite being able to classify Tor and non-Tor traffic with very high accuracy. In summary, the key contributions in this paper are as follows:

- We present an approach for classifying Tor and non-Tor flows, and identifying the Tor traffic type using deep neural networks (DNN). We perform experiments to evaluate the DNN models and compare their performance to previous models evaluated on the same dataset (UNB-CIC Tor network dataset).
- We show that DNN exhibited better accuracy (99.89%) than the previous models. Furthermore, in classifying Tor network traffic into respective types, the performance of DNN exceeds that of previous work by 6.2%.
- Finally, we evaluate the robustness of the DNN models on adversarial examples generated with a Generative Adversarial Network (GAN) model. Using the synthetic adversarial examples, we show that the DNN models initially fail to predict the packets generated by the GAN. However, on further retraining, the accuracy of predicting the adversarial examples improves.

The rest of this paper is structured as follows. Section II provides a brief introduction to the *Tor* network and discusses related work. Section III presents the methodology, while Section IV discusses the experiments. In Section V the GAN model architecture is described; while in Section VI results from different DNN architectures are presented and compared to results from related works. Section VII describes the adversarial attack on the models. Finally, Section VIII presents conclusion and future work.

II. Background and Related Work

A. Tor

The Tor network provides anonymity by bouncing traffic over a chain of intermediate computers. Generally, traffic is relayed through three types of nodes (a) entry/guard relay, (b) middle relay, and (c) exit relay (see Figure 1). Each of these relay nodes has specific roles:

- Entry/Guard relay: as the name indicates, this is the entry point of a Tor network and is selected from a directory server (collection of onion routers).
- Middle relay: are the intermediate relay nodes present between the guard and exit nodes. It acts as shield to prevent the entry and exit nodes from knowing each other.
- Exit relay: are the exit point of the Tor network. The exit relay node transmits data to the intended destination.

Tor minimizes the risk of traffic analysis by distributing communication over diverse places on the internet. Instead of considering a static route between the source and the destination, the Tor network randomly chooses a pathway so that no interceptors can reveal where the packets originated and where they are destined for. Tor incrementally constructs a private path by developing an encrypted circuit. This circuit is progressively created one hop at a time with each relay node only aware of where the data was received and to which node the data is to be forwarded. Moreover, each pair of communication is encrypted with separate Diffie Hellman keys, to ensure non-traceability of the communication path, thereby achieving anonymity.

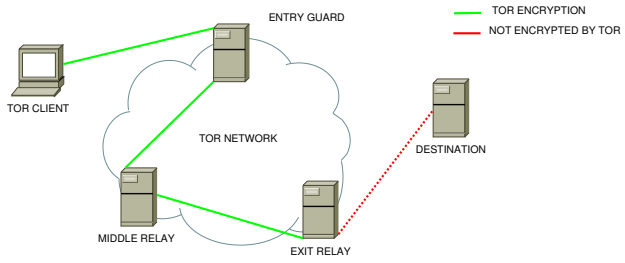

Fig. 1. A basic architecture of the Tor network

B. Related Work

Analysis of Tor traffic has been an active area of research in the last couple of years. Previous works such as [22], [23], [28], [2] are based on analysis of Tor Traffic within a Tor node. By contrast, [27] and [21] address the problem of characterizing Tor traffic through observing the encrypted network traffic between the client and entry node. In [21], the authors focused on detecting Tor from non-Tor traffic and identifying the Tor traffic type using 23 time-based features. The exclusive use of time-related features was to expedite efficiency and to ensure an encryption independent traffic classifier. We adopt a similar approach in this paper and base our experiments on their published dataset from real Tor and non-Tor traffic. However, different from [21] where traditional machine learning techniques such as ZeroR, C.45 and KNN were utilized, our classification and identification approaches are based on deep neural networks. Furthermore, we have analyzed the robustness of our approach to adversarial attacks using a Generative Adversarial Network (GAN).

In [20], the authors develop a machine learning approach for detecting *non-Tor* traffic in a network (implemented for an *Intrusion Detection System*), using the dataset generated in [21]. They propose two classifiers, one based on Artificial Neural Network (ANN) and another on Support Vector Machine (SVM). Moreover, they present a feature selection algorithm that allows them to select the most important features from the total number of features in the dataset. Finally, they show that an almost perfect detection accuracy of *non-Tor* traffic is indeed achievable.

In [1], the authors propose a system to detect and block *Tor* traffic in a network using deep packet inspection (DPI). In the technique presented, Tor traffic characteristics is obtained from deep packet inspection of the three-way handshake

packets exchanged at the connection establishment stage (i.e. TLS handshake process). Within the process, some signatures are extracted that distinguish the TLS handshake of Tor browser from the TLS handshake of ordinary browsers. These signatures are then used to detect Tor-bound traffic using Bro-IDS. Once Tor traffic is detected, the destination IP of the detected packet is logged and a proxy will use this information to block traffic to that destination. Unlike [1], our approach uses time-based features rather than DPI to detect Tor traffic. Moreover, unlike our study, the authors of [1] did not extend their work to detection of Tor Traffic types. It is also important to note that DPI cannot be applied to encrypted traffic.

In [24], the feasibility and effectiveness of attacks against Tor using NetFlow data was investigated. An active traffic analysis technique based on perturbation of the user traffic characteristic at the server side, while observing a similar perturbation at the client side through statistical correlation was applied. The method yielded 100% accuracy in revealing the actual source of anonymous traffic from in-lab testing. However, with real-world experiments an overall accuracy of 81.6% and false positive of 5.5% was achieved. Different from [24], our goal in this paper is classification of Tor traffic rather than de-anonymization. Also, our study is based on passive traffic analysis rather than active analysis that requires eavesdropping in real-time.

In [25], the authors used a simulated network to detect Tor traffic based on the use of packet sizes as features. They captured data in the form of 1MB capture files and recombined them with mergecap and then processed the files with NetAI to produce ARFF format files for use by the Weka machine learning environment. Random Forest, J48 and Adaboost algorithms were used to classify HTTPS and HTTP over Tor traffic. Random forest was ble to classify HTTP over Tor with 93.7% accuracy and a false positive of 3.7%, but classified HTTPS over Tor with 97.7% accuracy and 0.3% false positive. By contrast, our work classifies Tor traffic into 8 different types whilst also identifying Tor traffic from non-Tor traffic with high accuracy using deep learning models.

In [27], the authors used a *Hidden Markov Model* to separate encrypted *Tor* traffic into four different application types (P2P, FTP, IM and Web, while anything else is classed as unknown), thus detecting the type of encrypted traffic with a maximum accuracy of 92%. As features, they use burst volumes and directions extracted from Tor flows in contrast to time-based features used in this paper.

The study in this paper, unlike the aforementioned previous works develops DNN models for classifying Tor and non-Tor traffic, as well as identifying the traffic type from the encrypted Tor flows. Furthermore, unlike previous works, we test the robustness of the DNN models against adversarial examples of Tor traffic generated using GAN.

III. Methodology

This section describes the methodology employed for traffic classification in this paper. Here, we discuss the dataset, feature selection methods, classification algorithms, and evaluation metrics. Finally, we present the methods used to generate adversarial samples using Generative Adversarial Network.

A. Dataset Description

In order to build our machine learning models, we used the UNB-CIC *Tor and non-Tor* real-world dataset [21]. This dataset was generated by setting up three users to initiate activities using the Tor network and storing the packets in *.pcap* files. The traffic was collected using Wireshark and tcpdump resulting in a 22GB dataset consisting of 8,044 samples labelled as TOR while 59,784 were labelled as non-TOR. Further processing produced 28 features summarized in Table I. These features were generated by grouping packets by {*sourceIP, sourcePort, destIP, destPort, protocol*}. Two *.csv* files were produced with the generated features. In the first file, the features were labeled as *Tor* and combined with ones derived from normal traffic [26], labeled as *non-Tor*.). In the second file, the dataset includes eight types of Tor traffic generated for diverse applications (see Table II).

TABLE I
ATTRIBUTES EXTRACTED FROM NETWORK FLOW

Source IP	IP address sending the packets
Source port	Port used by the source
Destination IP	IP address receiving the packets
Destination port	Port used by the destination
Protocol	Protocol used for communication
Flow duration	Length of the connection (seconds)
Flow bytes	Number of bytes sent
Flow packets	Number of packets sent
Flow IAT	Packets flow inter arrival time (max, min, mean, std)
Fwd IAT	Forward inter arrival time (max, min, mean, std)
Bwd IAT	Backward inter arrival time (max, min, mean, std)
Active	Seconds in which the flow has been active (max, min, mean, std)
Idle	Seconds in which the flow has been idle (max, min, mean, std)

B. Feature Selection

In the preprocessing phase of machine learning (ML) pipeline, we use feature selection methods to eliminate irrelevant attributes. Typically, feature selection algorithms are broadly categorized into three groups (a) filter, (b) wrapper, and (c) hybrid approaches. Filter methods use intrinsic properties of attributes (i.e., distance, entropy, dependency or consistency) to determine if the subset of attributes represent the characteristics of observations. Wrapper approaches use accuracies of ML algorithms to evaluate the importance of selected features. However, the outcome of the selected subset of attributes is always biased towards the ML algorithm of choice. Hybrid methods combine the benefits of wrapper and filter methods to select an optimal subset of attributes. In this study, we use Correlation Based Filter Selection (CFS) [31] and Symmetric Uncertainty (SU) [31], two well-known feature selection methods to obtain the

TABLE II
DESCRIPTION OF TOR NETWORK TRAFFIC

Traffic Type	Description
Web browsing	HTTPS and HTTP traffic generated with Firefox and Chrome
Email	Mails were delivered through SMTPS and received using POP3S and IMAPS
Chat	Traffic samples were generated using instant messaging applications like Facebook, Hangouts, Skype, ICQ and AIM
Streaming	Youtube and Vimeo services were used to generate continuous streams of data.
File Transfer	The traffic was generated using services like FTP over SSH (SFTP), FTP over SSL (FTPS) and Skype.
VoIP	Voice traffic generated using Facebook, Skype and Hangouts.
P2P	Traffic generated from Torrent i.e., μtorrent and Bittorrent

optimal feature set. Before feature selection, we preprocess the dataset. In particular, we remove irrelevant attributes like *sourceIP, destIP and protocol* which were too specific for training a generic neural network. Moreover, we eliminated the rows consisting of NaN for all the attributes. In the following paragraphs, we introduce the feature selection methods used in our experiment.

1) **Symmetric Uncertainty (SU)** Symmetric Uncertainty is based on the concept of information theory i.e., entropy which measures the uncertainty/unpredictability of a random variable. The entropy of a variable X is defined using Equation 1.

$$H(X) = -\sum_{x \in X} P(x)\ \log_2(P(x)), \tag{1}$$

Now, if the entropy of X is partitioned with respect to another variable Y, then the entropy of X after observing Y is defined using Equation 2.

$$H(X|Y) = -\sum_{y \in Y} P(y) \sum_{x \in X} P(x|y)\ \log_2(P(x|y)), \tag{2}$$

Here, $P(x)$ and $P(x|y)$ are prior and posterior probabilities respectively. The decrease in the entropy of X with respect to Y gives additional information, and this is known as *information gain* defined as

$$IG(X|Y) = H(X) - H(X|Y), \tag{3}$$

It is important to note that information gain is biased towards attributes with larger values, and does not produce normalized values by which attributes can be compared. Thus, we used symmetric uncertainty (SU) to determine relevant features which can be defined by Equation 4.

$$SU(X|Y) = 2 \times \left(\frac{IG(X|Y)}{H(X) + H(Y)} \right). \tag{4}$$

SU, removes the bias introduced by information gain and produces values in the range of [0,1]. Here, a value of 1 indicates that the knowledge of one attribute can predict the importance of the other attribute, and a value of 0 indicates the independence of attributes X and Y. Eventually, eleven features i.e., `'source_port'`, `'dest_port'`, `'flow_dur'`, `'flow_iat_std'`, `'flow_iat_min'`,`'fwd_iat_mean'`, `'fwd_iat_std'`,`'bwd_iat_mean'`,`'bwd_iat_std'`, `'bwd_iat_max'`, `'bwd_iat_min'` are retained by applying symmetric uncertainty.

2) **Correlation Based Filter Selection (CFS):** A feature is considered important if it is *relevant* to a class and *redundant* to other features of the same class. In other words, relevance is the indication of the predictive power of an attribute for identifying examples of a class. Normally, relevance is determined by either estimating *linear correlation* or *entropy*. Finally, from the application of CFS with a fixed threshold, four attributes (i.e.,`'source_port'`, `'dest_port'`, `'bwd_iat_mean'`, `'bwd_iat_std'`) yielding high accuracy were obtained. CFS utilizes symmetric uncertainty for estimating the relevance of an attribute. Features are selected using a two step approach. In the first step, symmetric uncertainty ($SU_{i,C} > \delta$) of feature $f_i \in F$ with class C for a threshold δ is estimated. Subsequently, a list of attributes, arranged in descending order of symmetric uncertainty value is created. In the next phase, the goal is to identify *set of prominent attributes ($F' \subset F$)* such that there exist no attribute j such that $SU_{j,C} > SU_{i,C}$.

C. Machine Learning Models

In this paper, deep neural network models of various configurations were implemented and trained using 25 features out of the 28 (i.e. excluding *sourceIP, destIP and protocol*) described in Table I. A deep neural network consists of multiple layers stacked to form a more in-depth architecture [30]. An artificial neural network tries to learn the mapping between the features (*given as input*) and the labels (*given as output*). Typically, the neurons (non-linear units) are responsible for mapping the input to the output. Each neuron in the first layer takes its input from the feature vector and computes an output value that gets transferred to the next layer.

In the training phase, the weights in each neuron are adjusted using backpropagation and gradient descent algorithm to minimize error. The main difference between a classic artificial neural network and a deep neural network is the depth. In the latter, multiple hidden layers are present, thus increasing the performance of the model. Moreover, it is usually harder to train deep neural networks as fast as classic ones, so in recent years, much research has been focused on improving training times [19].

D. Evaluation Metrics

In this study, we evaluate the performance of the models using the following well-known metrics: Precision, Recall,

F-measure, Accuracy and Area Under Receiver Operating Characteristic Curve (AUC-ROC). The performance measures are shown in Table III. These metrics help us to validate the performance of the model on new/unseen data.

TABLE III
DESCRIPTION OF THE USED METRICS

Metric	Description
Precision (P) = $\frac{TP}{TP+FP}$	Measures how many samples tagged as Tor are indeed Tor.
Recall (R) = $\frac{TP}{TP+FN}$	How many true Tor applications are identified by the model from the total samples.
F-Measure = $\frac{2RP}{R+P}$	Harmonic mean of recall (R) and precision (P).
Accuracy (A) = $\frac{TP+TN}{TP+TN+FP+FN}$	Fraction of predictions that were correctly obtained from model.
ROC-AUC	ROC compares TPR (Recall) vs False Positive Rate (FPR), and AUC is the area underneath the entire ROC curve, with unity being the highest possible value.

IV. EXPERIMENTS AND RESULT ANALYSIS

In this section, we discuss the results of our experiments and demonstrate the effectiveness of our classification model. All experiments were performed using a system installed with Ubuntu 16.04.2 LTS operating system, Intel core i7 consisting of 16GB RAM. The DNN model implementation was carried out using the *Keras* library running a *Tensorflow* backend. Tensorflow supports both CPU and GPU device types. We particularly configured the system using *GPU* support for accelerating the training process.

We considered four scenarios in the experiments based on the network traffic dataset collected from [21]. The dataset is explained in detail in Section III-A. Besides, we created a classification model using network topologies shown in Table IV. Experiments, along with the goals, are described below:

- Scenario I: Identification of Tor and non-Tor traffic utilizing the DNN models. *The goal is to identify whether or not a given TCP flow is transmitting Tor traffic*.
- Scenario II: Performance evaluation of classification model to label a traffic into a specific category (i.e. browsing, email, chat, audio, video, file transfer, p2p and VoIP). *The goal is to identify the type of traffic flow existing within an encrypted Tor session*.
- Scenario III: We compare the accuracy of shallow network and the deep neural network.
- Scenario IV: Finally, we also evaluate the robustness of the classification model using an adversarial attack. In particular, we generate instances using Generative Adversarial Network (GAN). Subsequently, we assess whether the trained classifiers could identify the traffic generated with GAN.

A. Scenario I: Tor vs non-Tor

In this experiment, the DNN-A and DNN-B models (refer to Table IV) were trained using Tor and non-Tor traffic. To train the models, we used 70% of the traffic. Out of the remaining samples, 10% was used for validating the model while 20% was used as the test set. Primarily, the focus of this experiment was to investigate the capability of the classifiers to label traffic as Tor or non-Tor. Specifically, this scenario can be considered to be a binary classification task. To avoid over-fitting, we continuously retrained the classification model until the accuracy on the validation set improved. Table V summarizes the performance in the binary classification task. We observe that improved outcome is obtained with DNN-B with five layers, trained on 25 attributes (i.e. excluding *sourceIP, destIP and protocol* from the original 28 features outlined in Table I). DNN-B had an overall accuracy of 99.89%, while that of DNN-A was 98.81%.

B. Scenario II: Multiclass classification

In this experiment, given a *TCP flow* already recognized as *Tor* traffic (as in Experiment I), we intend to classify an unseen/new traffic into one of eight classes (*browsing, email, chat, audio, video, file transfer, VoIP, p2p*). As the scenario mentioned above depicts, in the case of multi-class classification we employed DNN-C, having eight neurons in the output layer. The evaluation was performed in a similar manner to the first scenario, splitting the dataset into three subsets: training set (75%), validation set (10%) and test set (15%). An F-measure of 0.95 along-with 95.60% accuracy is obtained with DNN-C for this experiment (see Table VI).

C. Scenario III: Evaluation on Synthetic Traffic Test Set

In this section, we perform a further evaluation of the DNN models using a separate batch of new test traffic to emulate an unknown traffic classification scenario. To obtain new test traffic, we used the *Shadow network simulator* with *Tor plug-in* [29] to generate new *.pcap* files; these were then processed in the same way as the original ones used for the dataset. Although Shadow simulates the network layer, it links to and runs real Tor software. Thus, Shadow allows us to run a private Tor network on a single machine.

Table VII shows the performance of the models on the test set derived from the Tor traffic generated using Shadow. Both DNN models can be seen to have performed quite well on the new test set. However it can be observed that the performance obtained with DNN-B is better than DNN-A. The overall accuracy for DNN-A is 97.13%, while that of DNN-B is 99.47%. The results are consistent with those obtained in experiment 1 (shown in Table V). Both sets of results show that the deeper the network, as in the case of DNN-B, the better the outcome of the classification. Deeper networks learn discriminant patterns to effectively differentiate observations of distinct classes.

TABLE IV

ARCHITECTURE OF THE PROPOSED MODELS

Architecture name	Number of layers	Neurons	Activations
DNN-A	3	[50, 25, 1]	[ReLu, ReLu, Sigmoid]
DNN-B	5	[200, 100, 50, 25, 1]	[ReLu, ReLu, ReLu, ReLu, Sigmoid]
DNN-C	4	[25, 50, 100, 8]	[Leaky ReLu, Leaky ReLu, Leaky ReLu, Softmax]
GAN generator	4	[25, 50, 100, 25]	[Leaky ReLu, Leaky ReLu, Leaky ReLu, Tanh]
GAN discriminator	4	[100, 50, 25, 1]	[Leaky ReLu, Leaky ReLu, Leaky ReLu, Sigmoid]

TABLE V

RESULTS FOR EXPERIMENT 1: CLASSIFYING TRAFFIC INTO TOR AND NON-TOR USING 25 FEATURES

Metrics	DNN (A)	DNN (B)
Precision (*Tor*)	99.15%	99.88%
Recall (*Tor*)	99.50%	99.99%
F-Measure (*Tor*)	0.99	0.99
Accuracy	98.81%	99.89%
ROC-AUC	0.99	0.99

TABLE VI

RESULTS FOR EXPERIMENT 2: CLASSIFYING TRAFFIC INTO THE EIGHT DIFFERENT CATEGORIES USING 25 FEATURES

Metrics	DNN (C)
Precision	95.68%
Recall	95.59%
F-Measure	0.95
Accuracy	95.60%
ROC-AUC	0.94

TABLE VII

RESULTS OF EVALUATING TRAINED MODELS ON A TEST SET OF SYNTHETIC TRAFFIC

Metrics	DNN (A)	DNN (B)
Precision (*Tor*)	96.84%	98.63%
Recall (*Tor*)	98.20%	99.83%
F-Measure (*Tor*)	0.97	0.99
Accuracy	97.13%	99.47%
ROC-AUC	0.97	0.99

V. GENERATIVE ADVERSARIAL NETWORK

We implemented the Generative Adversarial Network (GAN) by combining the ideas behind all three previous models. The resulting architecture was then simplified since the training process for a GAN is longer and more difficult to stabilize than a standard deep neural network. Therefore, using a simpler model accelerated the experimentation. The model architecture is shown in Figure 2. Note that this architecture was only implemented for the *Tor/non-Tor* dataset and is not directly applicable to the multi-class problem (i.e. traffic classification).

A Generative Adversarial Network[17] (GAN for short) is a special architecture deriving from the more classic deep neural network. In this setup, two neural networks are trained in a round-robin style. The two networks are called generator and discriminator, with the following duties:

- Generator: takes a random noise as input and tries to learn the probability distribution of the underlying data $P(x|y)$, generating fake samples that get passed to the discriminator and should fool it into predicting that these examples are real.
- Discriminator: takes as input both real and fake samples, originating from the dataset and the generator respectively. It aims to determine whether the given example is real or fake, and thus learn which values for which features make up a real sample or a fake one.

This approach is based on game theory, and from experimental results in different contexts, the trained model is usually more robust to external manipulation and less prone to overfitting [18].

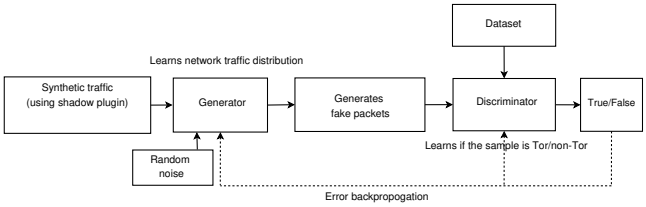

Fig. 2. Generative Adversarial Network for identifying network traffic

VI. COMPARATIVE ANALYSIS

In this Section, we compare the performance of our method with prior research work [20] and [21] that utilized the same dataset used in our experiments. Primarily, we compared the results for both binary classification problem and traffic type classification. The results of the ANN experiments are presented in Table VIII, while direct comparison with related works (namely [20] and [21]) are given in Table IX.

In particular, we implemented three ANN models which were reported in previous work. The first one is trained on 25 features, while the second (ANN-CFS) and third (ANN-SU) are trained on the dataset after applying the CFS and SU feature selection methods respectively. From our experiments, ANN had 98.09% accuracy and F-measure of 0.98, which outperformed the other two models (see Table VIII). These results show that both DNN-A and DNN-B models performed better than all three ANN models that we implemented.

Finally, we compare the best results obtained in our experiments with the best results from previous works [20] and [21] in Table IX. Note that these papers used the same dataset that we experimented with in this paper. From the tabulated results, we observe that the proposed model (DNN-B) outperforms the artificial neural network (ANN-CFS) in

identifying Tor traffic. Furthermore, in classifying Tor traffic into respective types, it can be seen that the performance of our DNN-C model is significantly better than the C4.5 and ANN-CFS models used in previous work.

TABLE VIII
RESULTS FOR TOR VS NON-TOR ON OUR IMPLEMENTATION OF ANN MODELS EMPLOYED IN PRIOR WORK.

Metrics	**ANN**	**ANN-CFS**	**ANN-SU**
Precision (*Tor*)	98.69%	96.18%	97.86%
Recall (*Tor*)	99.16%	97.83%	98.77%
F-Measure (*Tor*)	0.98	0.96	0.98
Accuracy	98.09%	94.66%	97.15%
ROC-AUC	0.99	0.96	0.98

TABLE IX
RESULTS COMPARISON BETWEEN OUR DNN MODELS AND THE BEST RESULTS FROM PREVIOUS WORKS [20] AND[21]

Experiment 1- Tor and non-Tor			
Metrics	DNN (B)	ANN-CFS [20]	C4.5 [21]
Precision (*Tor*)	99.88%	99.8%	94.8%
Recall (*Tor*)	99.99%	98.8%	93.4%
F-Measure (*Tor*)	0.99	0.99	-
Accuracy	99.89%	99.80%	-
Experiment 2- Traffic Type Classification			
	DNN (C)	ANN-CFS [20]	C4.5 [21]
Precision	95.68%	88%	79%
Recall	95.59%	79.4%	79%
F-Measure	0.95	0.84	-
Accuracy	95.60%	89.4%	-

VII. ADVERSARIAL ATTACK ON THE DEEP NETWORKS

As seen in the previous section, the detection of *Tor* traffic in a network is quite accurate, allowing the network owner to completely block *Tor*. Moreover, since the proposed models rely on low-level features of the connections, it is hard to escape detection. A possible countermeasure to this is to use the *generator* part of the Generative Adversarial Network to forge parameters that are close to those observed with *Tor*, so as to fool the detection system into thinking that the generated flow is *non-Tor* traffic.

In order to test this hypothesis, the best generator from all the training epochs was fed random noise to generate fake parameters. The generated features were then used as input to the previously discussed trained models and their responses determined. The results of this process are shown in Table X. These results show that DNN-B detected only a fraction of the faked Tor samples, i.e. 1.03%. On the other hand DNN-A was bypassed by all the adversarial samples, classifying 100% of them as non-Tor. This experiment also confirms that a GAN is usually more attack-resistant than classical DNN, as shown in Table X. These results are based on 1000 generated fake instances.

TABLE X
PERFORMANCE ON ADVERSARIAL ATTACKS

Models	**DNN (A)**	**DNN (B)**	**GAN**
% of forged examples classified as *non-Tor*	100%	98.97%	73.02%

The results confirm the validity of the formulated hypothesis, as the fake parameters generated initially deceived DNN-A in all cases and DNN-B in almost all cases. However, as shown in Figure 3, retraining with the adversarial samples improved the performance of DNN-B, but DNN-A misclassifies samples even when the retraining steps were increased. However, beyond 20 epochs, the number of misclassified instances drops sharply for DNN-B, while at 40 epochs and beyond, the misclassification of the adversarial samples is reduced to nearly zero. On the other hand, for DNN-A, it takes more than 60 epochs of retraining for the misclassifcation rate to drop from 100%; the misclassification rate gradually drops until epoch 120 and beyond where it stabilizes at about 8%. It is therefore quite clear from the figure that DNN-B (the deeper network) is more resilient to the adversarial attack than DNN-A.

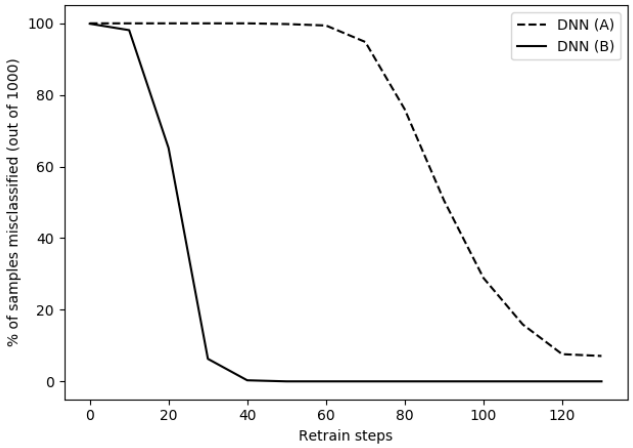

Fig. 3. Performance improvement of DNN with retraining

VIII. CONCLUSIONS AND FUTURE WORK

In this paper we presented Deep Neural Network models for detecting *Tor* traffic in a network. The models were investigated using several experimental scenarios. The results show that it is indeed possible to detect Tor with near-perfect accuracy using DNN and, moreover, it is also possible to detect (and thus block) specific categories or types of Tor traffic with DNN. Additionally, we showed that adversarial examples of Tor generated using a GAN could bypass the DNN models that previously detected Tor traffic with high accuracy. However, further retraining with adversarial examples proved to be a viable mitigation to the adversarial attack, indicating that the deeper networks were more robust to the attack. For future work, other types of deep learning architectures for Tor traffic detection and classification will be implemented and evaluated. Their robustness to adversarial attacks will also be investigated.

REFERENCES

[1] Saputra, Ferry Astika, Isbat Uzzin Nadhori, and Balighani Fathul Barry. "Detecting and blocking onion router traffic using deep packet inspection." Electronics Symposium (IES), 2016 International. IEEE, 2016.

[2] AlSabah, Mashael, Kevin Bauer, and Ian Goldberg. "Enhancing Tor's performance using real-time traffic classification." Proceedings of the 2012 ACM conference on Computer and communications security. ACM,

[3] Nguyen, Thuy TT, and Grenville Armitage. "A survey of techniques for internet traffic classification using machine learning." IEEE Communications Surveys & Tutorials 10.4 (2008): 56-76.

[4] Dingledine, Roger, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Naval Research Lab Washington DC, 2004.

[5] Subba, Basant, Santosh Biswas, and Sushanta Karmakar. "A Neural Network based system for Intrusion Detection and attack classification." In Communication (NCC), 2016 Twenty Second National Conference on, pp. 1-6. IEEE, 2016.

[6] Moore, Andrew W., and Denis Zuev. "Internet traffic classification using bayesian analysis techniques." In ACM SIGMETRICS Performance Evaluation Review, vol. 33, no. 1, pp. 50-60. ACM, 2005.

[7] Nguyen, Thuy TT, and Grenville Armitage. "A survey of techniques for internet traffic classification using machine learning." IEEE Communications Surveys and Tutorials 10, no. 4 (2008): 56-76.

[8] Zhang, Qingchen, Laurence T. Yang, Zhikui Chen, and Peng Li. "A survey on deep learning for big data." Information Fusion 42 (2018): 146-157.

[9] Alom, Md Zahangir, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Mahmudul Hasan, Brian C. Van Esesn, Abdul A. S. Awwal, and Vijayan K. Asari. "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches." arXiv preprint arXiv:1803.01164 (2018).

[10] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012.

[11] Majumder, Navonil, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. "Deep learning-based document modeling for personality detection from text." IEEE Intelligent Systems 32, no. 2 (2017): 74-79.

[12] Lv, Y., Duan, Y., Kang, W., Li, Z. and Wang, F.Y., 2015. Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems, 16(2), pp.865-873.

[13] Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A. and Lloret, J., 2017. Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things. IEEE Access, 5, pp.18042-18050.

[14] S. Y. Yerima and M. K. Alzaylaee, "High accuracy phishing detection based on convolutional neural networks", Proc. 3rd International Conference on Computer Applications and Information Security (ICCAIS), pp. 19-21, Mar. 2020.

[15] W. Ali and A. A. Ahmed, Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting, IET Information Security, vol. 13, no. 6, pp. 659669, 2019.

[16] S. Y. Yerima and M. K. Alzaylaee, "Mobile Botnet Detection: A Deep Learning Approach Using Convolutional Neural Networks," 2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), Dublin, Ireland, 2020, pp. 1-8.

[17] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

[18] Salimans, Tim, et al. "Improved techniques for training gans." Advances in Neural Information Processing Systems. 2016.

[19] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. 2015.

[20] Hodo, Elike, et al. "Machine Learning Approach for Detection of nonTor Traffic." Journal of Cyber Security 6.2: 171-194.

[21] Lashkari, Arash Habibi, et al. "Characterization of Tor Traffic using Time based Features." ICISSP. 2017.

[22] Bai, X., Zhang, Y., and Niu, X. (2008). Traffic identification of tor and web-mix. In 2008 Eighth International Conference on Intelligent Systems Design and Applications,volume 1, pp. 548-551.

[23] A. Chaabane, P. Manils, and M.A. Kaafar(2010). Digging into anonymous traffic: A deep analysis of the tor anonymizing network. In Proceedings of the 2010 Fourth International Conference on Network and System Security, NSS 10, pages 167174, Washington, DC, USA. IEEE Computer Society.

[24] Chakravarty, Sambuddho, et al. "On the effectiveness of traffic analysis against anonymity networks using flow records." International conference on passive and active network measurement. Springer, Cham, 2014.

[25] Barker, John, Peter Hannay, and Patryk Szewczyk. "Using traffic analysis to identify the second generation onion router." Embedded and Ubiquitous Computing (EUC), 2011 IFIP 9th International Conference on. IEEE, 2011.

[26] Gil, Gerard D., et al. "Characterization of encrypted and VPN traffic using time-related features." Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016). 2016.

[27] He, Gaofeng, et al. "Inferring application type information from tor encrypted traffic." Advanced Cloud and Big Data (CBD), 2014 Second International Conference on. IEEE, 2014.

[28] Z. Ling, J. Luo, K. Wu, W. Yu, and X. Fu (2014). Towards Discovery of malicious traffic over tor. In IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, pages 14021410.

[29] Jansen, Rob, and Nicholas Hooper, *Shadow: Running Tor in a box for accurate and efficient experimentation*, No. TR-11-020, MINNESOTA UNIV MINNEAPOLIS DEPT OF COMPUTER SCIENCE AND ENGINEERING, 2011.

[30] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y. and Alsaadi, F.E., 2017. A survey of deep neural network architectures and their applications. Neurocomputing, 234, pp.11-26.

[31] Yu, L. and Liu, H., 2004. Efficient feature selection via analysis of relevance and redundancy. Journal of machine learning research, 5(Oct), pp.1205-1224.