Neural Computing and Applications (2019) 31:8021–8033
https://doi.org/10.1007/s00521-018-3595-x

ORIGINAL ARTICLE

Deep learning to detect botnet via network flow summaries

Abdurrahman Pektaş[1] · Tankut Acarman[2]

Received: 17 November 2017 / Accepted: 23 June 2018 / Published online: 21 July 2018
© The Natural Computing Applications Forum 2018

Abstract
Compromised computer systems on the Internet, namely botnets, receive commands and share information with their central malicious systems while executing frequent and common network activities. Former botnet detection methods such as blacklists and botnet's signature matching cannot timely and reliably discover evolving botnet variants. Analysis of botnet network communication flows can be used to discover behavior of botnets toward detection. A rich dataset constituted by both botnet and normal network traffic flow summaries can be used for training and testing purposes. Furthermore, neural networks along emerging parallelization computing tools and processors may improve classification statistical metric results in an efficient manner. A neural network built by a higher number of layers and its architecture enhances classification accuracy. In this paper, we present a combination of convolutional and recurrent neural network to identify botnets. To validate the effectiveness of the proposed method, we test and benchmark the proposed method with two publicly available datasets, which are CTU-13 and ISOT, involving both botnet and normal data traffic. We evaluate statistical metric results by tuning the neural network architecture and compare the results with respect to baseline classifiers. Our experiment results show that the presented deep network learning-based botnet detection method is reached at 99.3% level in accuracy and 99.1% in F-measure, respectively.

Keywords Network security · Network flow · Botnet detection · Machine learning · Network traffic modeling

1 Introduction

A botnet is defined as the network of bots, which are compromised computer systems or devices on the Internet. These bots are commanded remotely by a botmaster, which is also called command and control (C&C) server. Unintended malicious activities such as sending spam e-mails, phishing, click-fraud, distributed denial of service (DDoS) attacks against critical targets are executed via these bots without the authorization of their system owner. The impact of a botnet can be severe due to its spreading capacity of malicious software, and large-scale malicious attacks are targeted to steal safety-critical or liability-critical personal data. Due to the constantly evolving behavior of botnets, the former approaches such as blacklists and botnet signature matching fail to detect botnets [7]. Blacklists ensure accurate defense against known and emerging cyber threats. For instance, blacklists are publicly advertised in [23, 30], but botnets change frequently their network connections, their IP addresses or domain names to evade detection. The usefulness of blacklists is rather limited, and blacklist-based approaches are not reliable and timely security solutions at identifying a botnet. Analysis of packet payload to search and match characteristics of botnet data, namely botnet signature, is computationally expensive and not scalable with respect to the volume of data generated by botnets on a high-speed network. For instance, Botzilla [39], Rishi [19] and Snort IDS rules [16] identify C&C channels in network traffic using payload byte signatures belonging to a fairly small set of known botnets. But in case of using encrypted C&C protocol, the payload data become irrelevant and not meaningful to capture characteristics of payload patterns. Payload

✉ Abdurrahman Pektaş
a.pektas@ebebek.com

Tankut Acarman
tacarman@gsu.edu.tr

[1] Ebebek, Icerenkoy Mahallesi Degirmen Yolu Caddesi No:37 D:6, 34752 Ataşehir, Istanbul, Turkey

[2] Computer Engineering Department, Galatasaray University, Ciragan Cad., 36, Ortaköy, 34349 Istanbul, Turkey

analysis cannot be used for identification of encrypted or obfuscated C&C protocols.

Botnets can be deployed in a centralized, decentralized or hybrid fashion depending on their C&C architecture (see for instance [2]). A botnet uses different communication protocols such as IRC, HTTP, P2P and IM. The IRC-based centralized C&C structure was very popular and commonly used by botnet creators. Currently, the HTTP(S)-based centralized or decentralized approach is executed. Because this protocol is the most prominent and used by Internet applications. To complicate detection, the cyber criminals may blend normal web traffic into botnet traffic flow. Encrypted or obfuscated C&C communication protocols hinder traditional signature-based botnet detection approaches, which inspect contents of C&C messages and trigger an alert if a malicious pattern is discovered.

Bots must communicate with their botmaster (i.e., C&C) to receive commands, to update their status or to exfiltrate critical data. Connection activities of bots constitute a particular behavior. Even evasive bots have been discovered by modeling behavior of botnet communication activities [5, 17, 26]. Bots generate uniform communication patterns of flows during the procedure of communication with their C&C server. And botnet traffic can be discovered by extracting these communication traces involving the underlying patterns of communication activities.

In this study, our particular focus is on the extraction of statistical-based network flow features between two hosts such as duration, size of packets, standard deviation of packet size, and other related flow-based features. Without loosing of generality, the proposed approach is also applicable to detect botnets that use encrypted communication protocols. The proposed system extracts network flow features and does not inspect the payload data. Then, we evaluate the performance of the deep neural network trained by flow-based features to identify botnets. Furthermore, we discuss the flexibility of the deep learning approach at inferring the distinguishable knowledge from the network dataset.

We make the following contributions to botnet detection by using flow-based features and deep learning:

(i) We design a botnet detection method combining both network flow information and deep learning.
(ii) For feature extraction, we use a graph structure to represent network connections between hosts. Using a structured graph, we easily extract statistical-based network flow features to build deep learning model.
(iii) We perform extensive experiments on large-scale real-world network capture traces. We test and benchmark publicly available datasets. The experimental results show that our approach outperforms the existing results and achieves high detection accuracy and very low false positive rate.

The rest of this paper is organized as follows. The literature review is presented in Sect. 2. In Sect. 3, we review the fundamental concept of deep learning. In Sect. 4, we introduce our method and we describe the feature set. In Sect. 5, we present our experimental study subject to real-world large-scale network captures. Finally, we draw some conclusions and discuss future work.

2 Related work

Since the last decade, dynamic and static analysis of a malware has attracted a lot of attention; computationally efficient accurate identification has been targeted. System resources such as network connections, processes, windows registry and file operations have been monitored to identify malware samples [35]. Computationally efficient identification of n-grams, mining and searching n-gram over API call sequences is introduced to discover episodes representing behavior-based features of a malware [36]. To identify a botnet, signature-based methods inspecting payload data, machine learning (ML) algorithms extracting features from network traffic patterns generated by connectivity between C&C server and bots have been introduced.

Haddadi and Zincir-Heywood [22] compared the effectiveness of five different botnet detection methods. Two methods are signature-based, which are named BotHunter and Snort. The remaining methods apply ML algorithms (MLAs) over different feature sets involving packet payload-based and network flow-based features. This study assesses the efficiency of these methods while performing multi- and binary classification tests on a dataset of 25 public botnets. The authors investigated the features with a minimum amount of a priori information to increase the detection accuracy. The experimental results show that flow-based features, such as interarrival time features, are the most representative features to model botnet communication and the highest classification accuracy is reached by C4.5 algorithm. The feature selection to detect anomaly was elaborated by Drasar et al. [15]. The impact of flow-based features to detection accuracy enhancement is evaluated. ML presents a significant advantage of automatically extracting representative and distinguishable characteristics from the botnet dataset. They do not require any prior information about botnet traffic such as the communication protocol, the heartbeat interval (i.e., packets sent back and forth to bots to sustain connectivity between C&C server and bots).

Supervised ML methods are effective solutions for solving various classification tasks in different domains by building a classification model from labeled training dataset. Stevanovic and Pedersen [48] introduced a flow-based botnet detection method. A total of 39 flow features such as source port, destination port, standard deviation of packets size, flow duration, are used to model malicious traffic. The authors evaluated eight supervised MLAs including naive Bayesian classifier, decision tree, SVM (NB), Bayesian network classifier (BNet), logistic regression. For experimental study, they used a publicly available dataset called ISOT [40]. According to the evaluation results, random forest algorithm identifies botnets with 95.7% accuracy. Chen et al. [8] focused on detection of botnet in high-speed and complex network by using supervised MLAs. They combined flow- and conversational-based features to build classification model. The experiments were conducted by using well-known CTU-13 dataset and the performance of different MLAs was analyzed. Random forest algorithm is reached at 94% level in accuracy.

Kirubavathi and Anitha [25] introduced a botnet detection method based on behavioral modeling of network traffic using flow features and supervised ML methods. The features are extracted from the information provided by small size packets sent and received in a flow; for instance, the ratio of incoming and outgoing packets, the initial packet length, the ratio of bot-response packets versus total packets in a flow constitute the four features used in this study. Naive Bayesian classifier is reached at 99% accuracy and 96.9% F-measure. But however, four features may not adequately represent botnet patterns and reliability of detection results need to be evaluated by more representative knowledge and characteristics.

Nogueira et al. [32, 41] proposed a botnet detection relying on characteristic traffic patterns. They used artificial neural networks (ANNs) as classification method to differentiate malicious traffic patterns versus benign or normal. The proposed method can generate alarms and trigger security actions. However, these triggered actions are not autonomous and need to be approved by security experts. ANN is built as a simple feed-forward back-propagation network consisting of input, hidden and output layers. Input layer has a dimension of n, where n is the number of features to represent a sample. The output layer has one neuron whose output shows whether the input sample is a botnet or normal traffic. The number of neurons in the hidden layer is tuned to reach at more accurate classification results. The experiments are carried out by using licit traffic captures (e.g., HTTP, P2P file sharing, video streaming and Skype), and illicit traffic captures are generated by SubSeven rootkit. Botnet detection rate is 87.56%. The usability of the results is rather weak since the feature set used by ANN to detect botnet is not described and the size of each application categories in the dataset is not presented.

Guntuku et al. [21] integrated a Bayesian regularization model to pre-process features and to select the most representative feature set. Botnet samples are executed in a controlled environment, and their network traces are captured as pcap files. Network flows of the captured traces are used as a feature set excluding source and destination IPs and ports, which are assumed to be volatile. Overall, the botnet dataset includes 55,824 network flows. To discover the most influential and representative features, the information gain method is used. Totally, 15 statistical flow-based features are extracted. Although the size of benign traffic dataset and their categories is not given, the proposed method detects botnets with an accuracy of 99.2%. Oujezsky et al. [33] analyzed botnet's behavior to extract life spans of C&C communication. The dataset includes real-world and simulated botnet traffic. Features are extracted from network flows such as duration, IP address and port numbers. Qiu et al. [38] implemented active learning to detect unknown botnets. This study is focused on the bidirectional packet size sequence information in order to discover the features and to capture periodical beacon signals sent to the bots. Alejandre et al. [3] extracted the feature set from network flows by applying genetic algorithm. The feature set includes 19 statistical flow features, and detection accuracy is evaluated by C4.5 algorithm. In our recent study [37], we evaluated feature selection methods. Linear models penalized with the L1 norm (aka Lasso), recursive feature elimination (RFE) and tree-based feature selection (aka random forest feature ranking) are compared while experimenting on public botnet traces. We tested and benchmarked flow-based statistical feature set extracted from network flow and empirically examined the impact of these extracted flow features on botnet detection.

Torres et al. [51] applied recurrent neural networks (RNNs) to detect botnet by modeling the behavior of network communications as a sequence of time-varying states. The behavior model of network connection is built by analyzing the long-term network activities and extracting distinguishable characteristics and patterns from these connections. A behavior model is built for each flow according to 4-tuple; the source and destination IP addresses, the destination port and the protocol. Mainly, size, duration and periodicity (interarrival time) feature of each flow are used to assign a state symbol subject to 3-level rating (i.e., short, medium and high). And the behavior of network connection is represented by a string of symbols. Behavioral string is transformed into a feature matrix by using a one-hot encoder that maps each symbol with a binary vector. The proposed approach was evaluated

with two different real-world datasets with a particular attention given to the effect of imbalanced network traffic in terms of a number of samples belonging to botnet and normal traffic, and the length of the sequence. According to the evaluation based on a stratified tenfold cross-validation, 97% detection rate and 3% false positive rate are achieved with the first dataset. However, for the second dataset, the detection accuracy and false positive rate are 81 and 3%, respectively. The proposed method suffers from detection loss about 16% in comparison with respect to the results of the first experiment. Large variation occurs because the first dataset contains significantly different behaviors between botnet and normal traffic, whereas the second dataset does not. More clearly, the botnet and normal traffic of the second dataset largely present the similar behavior. For the interested readers, Stevanovic and Pedersen [47] presented a comprehensive overview of MLAs applied to botnet detection. The authors summarized the existing botnet detection methods in three main sections dedicated to algorithms, feature set, performance evaluation and limitations.

The flow-based features have been largely investigated since packet payload inspection is computationally expensive and inefficient subject to botnet traffic volume and scale. Alauthaman et al. [2] focused on the connection behaviors between C&C server and bots, and they defined six rules to select the desired packets in order to reduce the number of packets to be analyzed and to avoid the encrypted traffic. A total of 29 TCP features are extracted based on the connection period equal to 30 s. A classification and regression tree is proposed, the entropy impurity at a given node is applied to determine the next node to be visited, and the ReliefF algorithm is applied to discover the impact of attributes subject to their values that discriminate between the instances near to each other. Features ranked by these two feature selection algorithms vary largely, and different subsets of features are identified. The ISOT dataset and the ISCX dataset are used for evaluation purposes, and the presented approach detects botnets with 98.32% accuracy, 98.69% F-measure and 0.75% false positive rate. Computation time for feature selection is 2.5 s and shorter with respect to the ReliefF algorithm. Sheikhan and Jadidi [45] leveraged flow-based detection to reduce the data size and processing time versus packet-based intrusion detection in high-speed links. ANNs and a multilayer perceptron neural architecture is used to detect untrained attacks, and a modified gravitational search algorithm (MGSA) was introduced to tune the interconnection weights of the neural anomaly detector. Dataset features are the number of packets per flow, the size per flow, flow duration, source and destination ports, TCP flags and IP protocol. By using test dataset, classification rate of flow-based anomaly detection is reached at 97.76% for the MGSA slightly higher than 97.71% for the particle swarm optimization (PSO) algorithm and 96.14% for the traditional error back-propagation (EBP) algorithm. Training time of the MGSA costs 2065 s, which is less than PSO's training time about 7136 s and considerably higher than the EBP about 98 s.

Recently, graph-based features and social network graph structure have been introduced in order to improve botnet identification. Wang and Paschalidis [53] introduced two-stage early botnet detection approach. At the first stage, flow- and graph-based anomaly detection methods are used. Both of these methods employ the theory of large deviations [43]. The flow-based approach quantizes network flow and examines the histogram of quantized data. Then, the anomaly detection approach creates graphs based on packet level data extracted from network traces and considers their distributions. At the second stage, social network graphs are created to characterize the interaction of bots. Then, community detection method is applied to capture highly interactive nodes. The detected node is labeled as a probable botnet. The proposed method is evaluated with the subset of the CTU-13 botnet dataset [18] and achieves 0.14, 0.088, 0.21, 0.082 and 0.14 F1-measure score for each scenario. Chowdhury et al. [10] considered seven graph-based features representing bot's connectivity. Then, a cluster of nodes, which are possibly bot candidates, is formed in order to fast detect bots. The cluster size is less than 100 nodes, which are less than 0.1% of all nodes in the network. But only CTU-13 dataset is used for testing.

In this study, we exploit network communication traffic between endpoints by representing the whole traffic in a graph. We extract the feature set generated by flow statistics between each host for 3 different protocols, namely TCP, UDP and ICMP. For instance, communication between each host is represented by these protocols as a matrix of summary network information. Therefore, any prior knowledge about botnet such as communication protocols (for instance, HTTP, IRC, P2P) and information about port or payload data are not required. We build a graph structure to represent network connections between hosts, and we derive statistical-based network flow features. Beyond the state-of-the-art, our particular focus is on the development of the long short-term memory (LSTM) and convolution neural network (CNN) to improve botnet detection system performance.

3 Deep learning

Deep learning refers to an artificial neural network (ANN) and is a sub-field of machine learning. The deep term becomes very popular in artificial intelligence and refers to the number of hidden layers in the neural network. ANN is

biologically inspired by the architecture and function of the human brain that learns from large-scale observational data. The usefulness and applicability of neural networks are enhanced by the introduction of CNN and RNN architectures. The CNN architecture is designed to manipulate and to encode certain hidden properties of input data, which are generally multi-dimensional array. The dimension of the input arrays can be adapted such as 1D for sequences of text, 2D for gray scale images and 3D for colored images or videos. CNNs are composed of three layers, which are convolution, pooling and fully connected layers. Convolutional layer applies convolution operation, i.e., it performs dot products between given filters and local regions of the input data and creates the convolved features. Fully connected layer is a former neural network layer, and it has connections to all units in the pooling layer. Pooling layer corresponds to gradual reduction of the spatial size of the convolved features and reduces computation cost and the size of parameters to be tuned.

RNN is mainly designed to extract semantical information from the sequential data. The traditional feed-forward neural network (including CNN) independently processes inputs in only one direction from the input layer to the output layer. However, for many tasks including sequential data such as text, speech or video, inputs and outputs are highly correlated. RNNs perform same operation for every element of a sequence and also for the output of the previous computations. For instance, the output of a layer is added to the following input and fed again to the same layer. Training RNN is costly due to its particular architecture. At each time index, RNN is equivalent to an entire layer in a feed forward network. From the computation cost perfective, training RNN for 500 times is similar to training a 500-layer feed-forward network simultaneously. Typical RNN architecture is illustrated in the left side of Fig. 1, and unfolded form into a fully connected network is plotted in the right side, where W presents the weight vector, X_t indicates the input, y_t is the output, and S_t is the hidden state at time index t. LSTM network has been introduced to reduce the complexity inherited by the architecture of RNN. Basically, LSTM acts as a dynamic caching and assists the network to decide when to ignore the current input and when to recall it. LSTM is a type of RNN that tracks gradient parameter optimization and

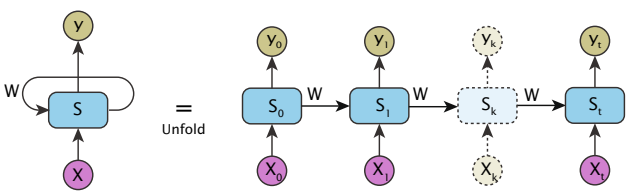

Fig. 1 Overview of the recurrent neural network architecture

remembers long-range dependencies. (The interested readers may investigate the chapter 9 and 10 elaborating CNN, RNN and LSTM in [20].)

4 Proposed methodology

In order to detect botnets, we propose a novel flow-based botnet discovery method and we deploy deep neural network to classify network traffic whether normal or botnet. As illustrated in Fig. 2, the proposed system consists of three major stages. The first stage is called feature extraction; it is responsible for extracting flow features from network traffic and transforming the extracted features into a multi-dimensional feature vector. The second stage is dedicated to building a classification model, and the input is the feature vector, whose elements are labeled whether normal or botnet. In our experiments, we evaluate deep neural network subject to the number of hidden layers and the number of units (neuron) in each hidden layer. At the final stage, we evaluate the accuracy of the different deep network architecture by using a set of manually analyzed and publicly available botnet traffic traces.

Bots connect to C&C servers on a regular basis to take commands. As a result, the amount of data transmitted between C&C and bot create the connection patterns, which can be used to identify botnet in raw network traffic. The main idea of the proposed method is to split the network traffic between endpoints and to represent these flows as a graph toward modeling the interaction and the behavior of connection. This graph-based model of host interaction is used to extract the feature set.

Input : NetFlow Records F
Output: Communication Graphs constructed by endpoints(nodes): (G)
foreach *flow recod*$(f) \in F$ **do**
  extract flow attributes such as protocol, duration, SrcAddr, etc.
  if G *has not node* $SrcAddr$ **then**
    add $SrcAddr$ node to Graph G
  end
  if G *has not node* $DstAddr$ **then**
    add $DstAddr$ node to G
  end
  if G *has edge between* $SrcAddr$ *and* $DstAddr$ **then**
    append extracted flow attributes to the edge
  end
  else
    create an edge between $SrcAddr$ and $DstAddr$
    append extracted flow attributes to the edge
  end
end

Algorithm 1: Pseudo-code for constructing commu-nication graph using network flow data

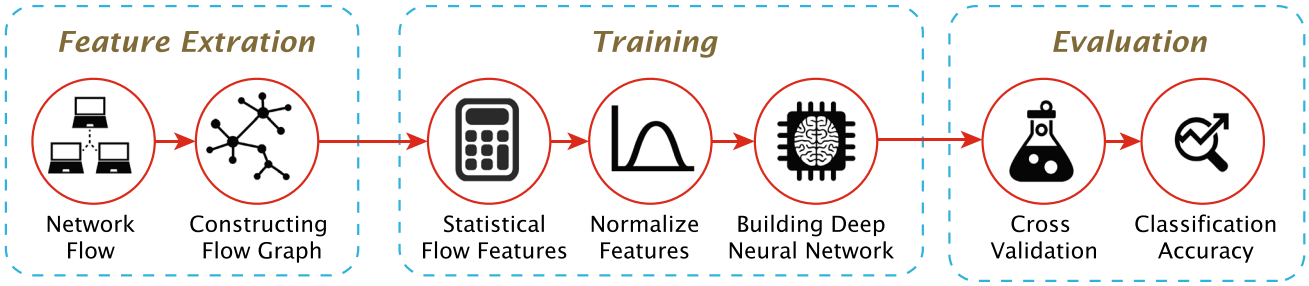

Fig. 2 Overview of the proposed botnet detection methodology

4.1 Feature extraction

The constructed flow graph is the core of feature extraction process, in which all flow data are treated and processed in order to extract the feature set. We compute different statistical features from the created flow graph. Pseudo-code for constructing communication graph using network flow data is given in Algorithm 1. Essentially, we iterate each flow record, and we add each source and destination IP of each connection and create an edge between these two nodes. Then, flow attributes such as duration, number of packets and bytes are affiliated with this edge. The extracted features are evaluated for the three protocols that are TCP, UDP and ICMP. For instance, Table 1 presents the extracted features from flow data using TCP protocol. Since these features are the same for both UDP and ICMP protocols, they are not repeated, and for the sake of clarity, TCP features are only shown in Table 1. Overall, TCP, UDP and ICMP protocols are considered while performing feature extraction from network flow data.

Example 1 Let us consider the network flow records shown in Table 2. These flows are obtained from the CTU dataset capture number 49. The network flow records are processed line by line. In the first line, flow attributes are extracted first and since the communication graph does not contain the source IP (147.32.84.165) and destination IP (147.32.80.9), these two nodes are inserted into the graph. Then, the edge is created between these two nodes by attaching flow attributes to the edge to represent communication. For the second flow record, the attributes are appended to the edge. In the fifth record, the destination IP (91.220.0.52) is added to the graph and flow attributes are attached to this created edge. After processing all network records, the communication graph is obtained as illustrated in Fig. 3.

One remark to note here is that the periodicity feature, which is the interarrival time, is calculated by subtracting the value of *Starttime* of the consecutive network flow and for the first entry of the communication, the periodicity value is set to 0. Following the creation of the communication graph, the features are easily computed by iterating all edges in the graph. Although numerical features are calculated by statistical functions, *state* features are appended into an array.

As shown in Table 1, the majority of the features are computed statistically over the array of network connection between two nodes (source address and destination address). Statistical derivation of feature gives standard deviation of the array of number, and each array constitutes at least 3 entries. Therefore, in our experiment, we only take into account the nodes that communicate at least 3 times over the same protocol, which is TCP, UDP or ICMP.

The feature called "*connection states*" is a string type, and it indicates the basic state of the communication in a flow record. Different states depend on the status of the transaction and the protocol type. For example, **CON** state indicates that the transaction is active. The state **URP** means "unreachable port," and this state denotes a particular behavior of the ICMP protocol. The communication states and their values are obtained by argus network flow extractor [1]. After feature extraction from network flow data, we transform connection states into the word embedding to capture sequential relations while constructing the deep learning architecture. However, for conventional machine learning (ML) methods the connection states are transformed into one-hot vector $V = 1, 0, 1, 0, ...,$ in which 1 indicates that the related connection state is active, whereas 0 indicates related connection is not active. These types of transformation are also used in natural language processing to model text-based features. If two nodes do not communicate over TCP protocol, then, all features are assigned by 0.

In summary, by following the extracted features from flow data using TCP protocol in Table 1 and the communication graph illustrated in Fig. 3, the role of the normalized graph features is to statistically model network flows. In addition, the role of the sequence of connection state is to reveal the intricate communication semantic. The sequence of connection state captures the behavioral

Table 1 Description of the feature set used in our study

Feature	Type	Number of feature	Short description
tcp_duration	Float	5	Mean, standard deviation, median, maximum and minimum values of TCP duration
tcp_bytes	Float	5	Mean, standard deviation, median, maximum and minimum values of TCP bytes
tcp_packets	Float	5	Mean, standard deviation, median, maximum and minimum values of TCP packets
tcp_periodicity	Float	5	Mean, standard deviation, median, maximum and minimum values of TCP periodicity
tcp_states	String	Number of states	TCP communication states
tcp_repeated_count	Integer	1	Total count of TCP communication between two nodes

Table 2 Sample of network flow records for Example 1

Start time	Duration	Protocol	Source address	Source port	Destination address	Destination port	State	Total packets	Total bytes	Source bytes
16:52:50.94	0.00	udp	147.32.84.165	1025	147.32.80.9	53	CON	2	203	64
16:52:53.20	0.00	udp	147.32.84.165	1025	147.32.80.9	53	CON	2	590	87
16:53:42.31	0.05	udp	147.32.84.165	1025	147.32.80.9	53	CON	2	226	71
16:53:47.62	0.55	udp	147.32.84.165	1025	147.32.80.9	53	CON	2	212	77
16:55:44.17	3.38	tcp	147.32.84.165	1165	91.220.0.52	80	FSPA_FSPA	11	1198	736
16:55:47.54	3.31	tcp	147.32.84.165	1268	91.220.0.52	80	FSPA_FSPA	11	1069	607
16:56:20.73	0.34	tcp	147.32.84.165	1275	91.220.0.52	80	FSPA_FSPA	10	1007	545
16:56:50.97	3.35	tcp	147.32.84.165	1280	91.220.0.52	80	FSPA_FSPA	13	1189	667
16:56:58.13	0.00	icmp	110.34.38.106	–	147.32.84.165		TXD	1	90	90
16:57:24.38	0.40	tcp	147.32.84.165	1286	91.220.0.52	80	FSPA_FSPA	10	1007	545
16:57:54.62	8.95	tcp	147.32.84.165	1292	91.220.0.52	80	S_	3	186	186
16:58:49.26	0.00	icmp	110.34.38.106	–	147.32.84.165		URFIL	1	70	70
17:00:25.61	0.00	icmp	110.34.38.106	–	147.32.84.165		URFIL	1	70	70
17:04:01.79	1.16	icmp	110.34.38.106	–	147.32.84.165	–	URP	3	210	210

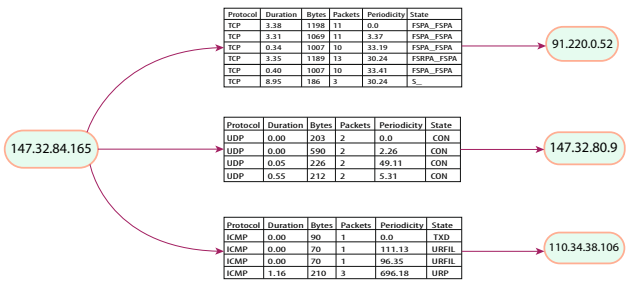

Fig. 3 The constructed communication graph for Example 1

communication patterns. For example, when performing SYN flood, the connection states include numerous SYN flag set flows or when exfiltrating data through ICMP protocol, there is a long sequence of connection state composed of URFIL. In consequence, we leverage the sequence of the connection state between two endpoints as a feature to discriminate botnet and normal traffic.

4.2 Architecture of deep neural network

The proposed deep network consists of four major parts: embedding, convolution, LSTM and fully connected networks. The overview of deep learning architecture is plotted in Fig. 4. From the sequence of connection states toward the output of the deep learning architecture, the first block is the embedding function, and it maps and associates a numeric vector with every connection state, which belongs to the connection array between two nodes. Embedding leverages the distance function calculating between any two vectors to capture the relationship between the two correlated connection states. The matrix formed by these vectors is called embedding whose size is 25×1000. The convolutional part includes one-dimensional convolution layer and max-pooling layer. On the one hand, the convolutional part extracts hidden features from network state's sequence represented by an embedding vector. On the other hand, convolution filter block extracts

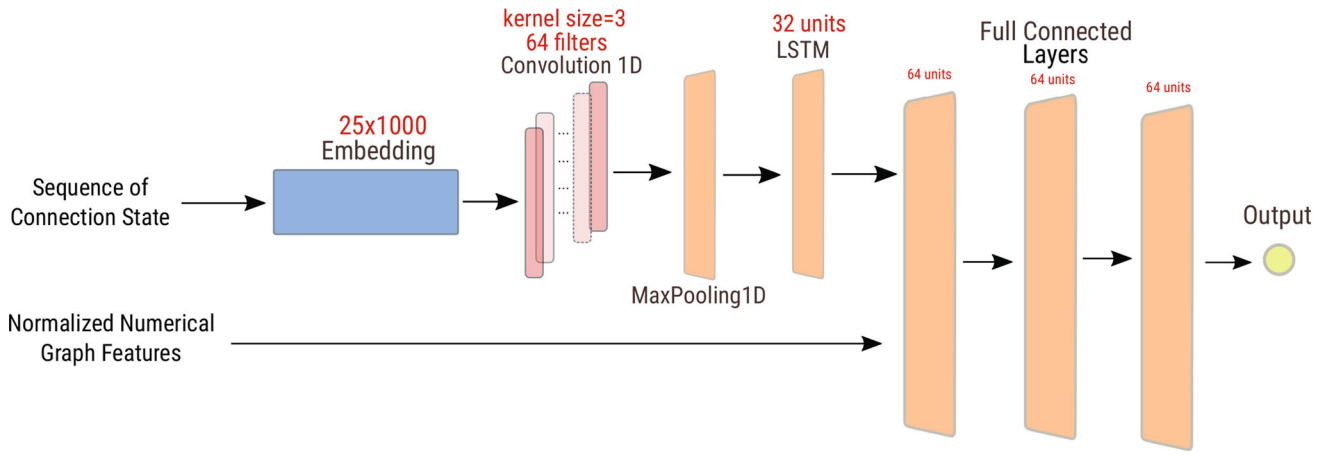

Fig. 4 The deep learning architecture for botnet detection

the relationship between input vectors and generates original features. We use convolution filter of size 3×64. After the convolutional layer, max-pooling is used to reduce the dimensionality of data. The outputs of the max-pooling layer are forwarded as an input to the LSTM block, which is a particular type of the RNN architecture. Then, we apply LSTM layers with 32 units to extract and model the network states' sequence. The LSTM layer explicitly enables capture of the sequential constraints. Then, the normalized statistical graph features and the output of the LSTM layer are merged and connected to the fully connected layers. We use three fully connected layers, and each consists of 64 units. Finally, the output layer predicts the label, e.g., it generates the probability of being whether botnet or normal by utilizing softmax activation function.

While building the model, we use dropout in all layers except embedding layer to avoid overfitting problem. Concerning the weight update procedure in a hidden layer, dropout randomly selects hidden units in the layer and neglects them, and the probabilistic approach is enforced. Before feeding the continuous numeric data into the deep network, we scale the input vector and normalize the input vector into range $(-1, 1)$. Hence, both training and testing time of the deep network converge faster.

5 Evaluation

We first describe the datasets, namely CTU-13 and ISOT, which are benchmarked by security researchers to test and evaluate their methods. Then, we define classification metrics such as recall, precision, F-measure and overall accuracy metrics to measure the effectiveness of our presented method. Finally, to empirically reach at the highest level in metric values, we combine the stand-alone network architectures and compare the classification results.

5.1 Dataset overview

The CTU-13 is a well-known public benchmark dataset for botnet research provided by the Czech Technical University. The dataset, which includes both botnet and normal traffic, consists of 13 different network traces belonging 7 botnet families. The brief description of the dataset including the capture duration, label of the capture, distribution of normal and botnet traffic for each capture is given in Table 3.

For evaluation purposes, we use another public botnet dataset called ISOT. The ISOT dataset combines several existing malicious and non-malicious datasets. It contains two different types of botnet: Zeus and Waledac. To normalize botnet traffic, non-malicious everyday network traffic such as web surfing, popular gaming and file sharing is incorporated into botnet traffic. The key information about botnet types in the benchmarking datasets is outlined as follows:

- **Neris** [31] botnet uses an HTTP-based C&C channel. The main activities are to send SPAM e-mails and to perform some click-fraud activities.
- **Rbot** [27] is IRC-based botnet. Rbot sample connects to the IRC channel to get information from botmaster.
- **Virut** [28] uses an HTTP communication channel. The main activities are to download executable files, to send SPAM e-mails and to intercept user inputs.
- **Menti** [46] is IRC-based botnet. The main activities are to employ a custom unencrypted protocol to connect to C&C server and to scan SMTP servers (i.e., TCP port 25)
- **Sogou** [13] connects to an unencrypted HTTP C&C channel, downloads some binary and compresses files.
- **Murlo** [12] is IRC-based botnet. The main activities are to download some executable files and to scan the local network ports.

Table 3 Overview of the dataset

Dataset name	Scenario ID	Duration (in min)	Botnet flows	Normal flows	Botnet name	# of bots	Communication channel
CTU-13	1	369	39,933	2,784,703	Neris	1	HTTP
	2	253	18,839	1,789,283	Neris	1	HTTP
	3	4010	26,759	4,683,879	Rbot	1	IRC
	4	253	1719	1,119,357	Rbot	1	IRC
	5	698	695	129,137	Virut	1	HTTP
	6	131	4431	554,488	Menti	1	IRC
	7	23	37	114,040	Sogou	1	HTTP
	8	1170	5052	2,949,178	Murlo	1	IRC
	9	311	179,880	2,574,004	Neris	10	HTTP
	10	285	106,315	1,203,476	Rbot	10	IRC
	11	16	8161	99,090	Rbot	3	IRC
	12	73	2143	323,328	NSIS.ay	3	P2P
	13	982	38,791	1,886,358	Virut	1	HTTP
ISOT	14	NA	78,754	2,743,258	Waledac	3	P2P
					Zeus	1	HTTP

- **NSIS.ay** [50] is P2P-based botnet. The activities are to access to several websites and to download some executable files.
- **Waledac** [24, 49] (also known as Storm) uses P2P communication channel. Waledac botnet is deployed to make money by sending spam e-mail, to download and to install malicious applications.
- **Zeus** [4] uses HTTP protocol to communicate between bot and C&C server. To evade payload inspection, payload data are encrypted by RC4 (Rivest Cipher 4) algorithm. Zeus botnet is mainly deployed to steal banking information.

The underlying communication parameters such as contents of the command, IP, port or host address used by botnets are presented in [17].

In our experimental study, we use connections based on TCP, UDP and ICMP protocols. The CTU-13 dataset is highly imbalanced in terms of number of connections. In other words, the volume of normal traffic exceeds the volume of botnet traffic, which represents the ground truth traffic on the Internet, and we can confirm that the dataset simulates the real-world botnet infection. In general, botnet traffic constitutes a small fraction of the entire network communication flows because botnet infection rarely occurs in a network. In this study, botnet traffic refers to the connection established between bot and C&C, in which data packets are sent and received, whereas normal traffic refers to all legal network traffic except botnet traffic.

5.2 Performance metrics

We evaluate the proposed method subject to 4 different metrics that are overall accuracy, precision, recall and F-measure. Reaching at the highest accuracy does not necessarily mean that the classifier correctly predicts with high reliability. Therefore, we use other measures to assess the reliability of the proposed system results. The performance metrics are defined based on the following definitions:

- *TP* refers to the correctly predicted botnet flow.
- *TN* is the number of the correctly predicted normal traffic flow.
- *FP* refers to the incorrectly classified botnet flow.
- *FN* refers to the number of incorrectly classified normal traffic flow.

In our study, botnet traffic refers to the positive label and normal traffic refers to the negative label. The precision is the proportion of true positives versus the sum of positive instances, more clearly, it is the probability for a positive sample to be classified correctly. The recall is the proportion of instances that are predicted positive and actually positive (i.e., *TP*). The F1-score, also known as F-measure or F-score, is the weighted harmonic mean of the precision and recall. F1-score is reached at its best value at 1 and its lowest value at 0. In the binary classification problem, the precision and recall contribute equally to F1-score. However, in multi-class evaluation studies, F1-score is calculated by taking the weighted mean of F1-score of each class. And, the overall accuracy is the proportion of total

number of correctly predicted instances over total number of instances.

5.3 Evaluation results

To evaluate the performance of the proposed method, we implement deep learning models by using Keras [9] Python library with the TensorFlow backend deep learning engine [11]. TensorFlow is an open-source framework available for mathematical computation using abstract data flow graphs. TensorFlow is developed by Google Corp. and mainly introduced for deployment of deep neural network. TensorFlow can be run on either a CPU or a GPU. Due to parallel processing capability, GPU drastically reduces processing time cost for training deep neural network.

We run our experiments on a workstation configured with 16 Intel Xeon E5 2600 v4 processor, 1TB hard disk, 64G RAM and two GPUs, namely GeForce 960 and the NVIDIA TITAN X GPU. We use tenfold cross-validation approach to validate the effectiveness of the proposed deep learning approach for botnet detection. Since benchmark datasets are imbalanced in terms of sample number in normal and botnet class, we adapt stratified K-Folds method in our evaluation study. Basically, stratified K-Folds validator splits the data into training and testing sets by preserving the percentage of samples for both classes.

We conduct extensive experiments with various neural architecture, including LSTM, CNN and fully connected networks in order to identify botnet traffic. By combining a LSTM, RNN and fully connected layers, we are able to obtain the best classification result as shown in Table 4. We also compare the proposed approach with standard machine learning algorithms (MLAs) used for botnet detection in the literature. These algorithms are random forest [6], support vector machine (SVM) [54], logistic regression [42, 55] and K-nearest neighbor algorithms [14]. We employ Scikit-learn ML toolkit [34, 44]. All these conventional ML approaches are executed on a CPU. And, we execute all deep learning tasks on the GPU and also on the CPU to compare run time cost. Furthermore, we analyze the performance of the stand-alone architecture of LSTM, CNN and dense layer constituting the proposed approach.

For different neural network architectures and standard ML classifiers, the classification results and the execution times are compared in Table 4. When these results are compared, the deep neural network, which constitutes by the combination of RNN, LSTM and fully connected layers (e.g., dense layer), is reached at 98.8 and 99.1% F1-score for ISOT and CTU-13 dataset, respectively. F1-score of this network is higher than any stand-alone deep network architecture. The stand-alone LSTM layer has more impact on the classification result improvement. Among standard MLAs, random forest classifier is reached at the highest F1-measure at 93.2 and 84.0% in 2.18 and 46.24 s for ISOT and CTU-13 dataset, respectively. When compared to the run time performance of each classifier, the deep learning methods cost less. As shown in Table 4, processing time is increased when the size and structure of the neural network are increased because we run them on the GPU to converge faster. When deep learning architecture is processed on the CPU, the execution time costs almost 14 times grater than the GPU average time. SVM and KNN algorithm convergence time is longer since these algorithms cannot handle large-scale data.

To interpret the representation learned by the combination of CNN and LSTM layers on the sequence of connection state feature, we use t-distributed stochastic neighbor embedding (t-SNE) [29, 52] to visualize the activation values of LSTM layer and last dense layer. t-SNE reduces high-dimensional vector to a lower-dimension vector representation. And we apply t-SNE to the sequence of connection state vector with the dimension of 32 to project to a two-dimensional vector space. The graph illustrating the activation values of LSTM layer is plotted in Fig. 5, which visualizes that CNN and LSTM learn partially meaningful and distinguishing feature from the connection

Table 4 Classification results with respect to different neural network model parameters

Algorithm	Processing time		Precision		Recall		F-score		Accuracy	
	ISOT	CTU-13	ISOT	CTU-13	ISOT	CTU-13	ISOT	CTU-13	ISOT	CTU-13
SVM	141.17	3980.69	0.871	0.842	0.83	0.782	0.850	0.811	0.867	0.801
KNN	62.06	2598.22	0.914	0.862	0.935	0.812	0.924	0.836	0.943	0.862
Random forest	2.18	46.24	0.922	0.864	0.942	0.817	0.932	0.840	0.952	0.872
Logistic regression	0.81	14.73	0.898	0.853	0.902	0.807	0.900	0.829	0.931	0.842
Stand-alone CNN layer	14.67	197.24	0.962	0.981	0.961	0.982	0.961	0.981	0.980	0.984
Stand-alone LSTM	18.93	251.12	0.971	0.988	0.969	0.984	0.970	0.986	0.984	0.986
Stand-alone dense	13.92	183.92	0.932	0.972	0.955	0.976	0.943	0.974	0.967	0.974
Combined layers	28.69	390.13	0.989	0.991	0.988	0.992	0.988	0.991	0.995	0.993

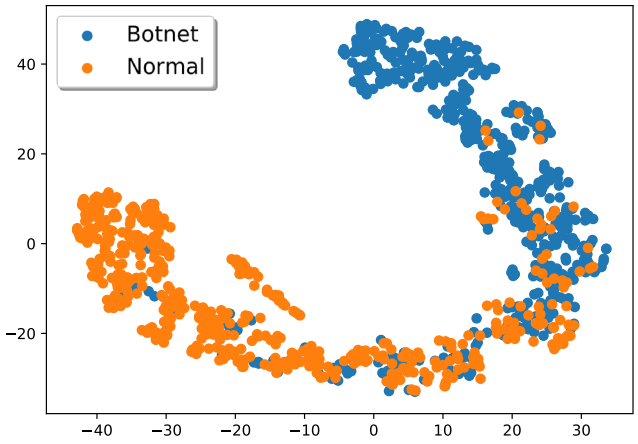

Fig. 5 Visualization of activation values of the LSTM layer for 1000 samples with their corresponding class

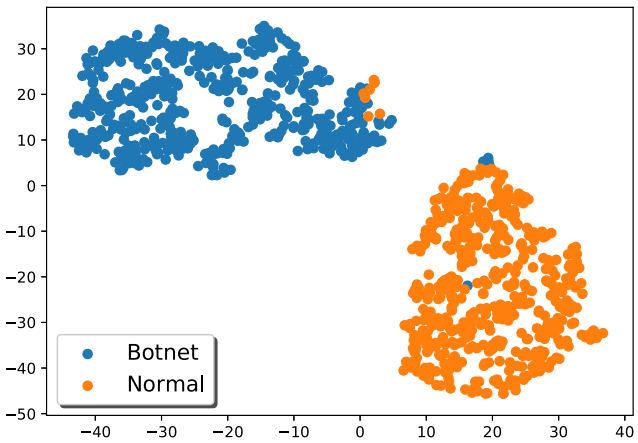

Fig. 6 Visualization of activation values of the last dense layer for 1000 samples with their corresponding class

state and differentiate normal and botnet traffic. To further investigate the performance of the overall deep learning architecture and the learned hidden features, we apply t-SNE to visualize the activation values of the last dense layer, (Fig. 6). Visualizations in Figs. 5 and 6 demonstrate that the proposed architecture better learns features and separates botnet and normal traffic more precisely.

The impact of each individual feature type on the deep learning performance is evaluated and tested. When the normalized graph feature is exploited only, detection accuracy and F-measure are reached at 92.2 and 91.3%, respectively. And, when connection state features are only used in the learning architecture, 71.4% accuracy and 70.2% F-measure are achieved. Thus, we can deduce the normalized graph features derived from flow statistics assure detection, but the connection state features do not provide sufficient semantical information toward modeling botnet traffic. When two types of features are fused, the resulting latent feature representation along with the deep network architecture leads to the highest botnet detection performance.

5.4 Comparison with respect to existing studies

Evaluation of same ML method with different datasets may give different results. For comparison purposes, the performance of different botnet detection approaches needs to be evaluated subject to the same dataset. Otherwise, comparison versus different methods is not reliable and accurate. We compare our results with respect to the state-of-the-art botnet detection methods using the CTU-13 and ISOT dataset.

Performance comparison of the presented system with respect to some existing studies for botnet detection is shown in Table 5. As stated in the related work section, Chen et al. [8] achieve 94% accuracy on CTU-13 dataset and Wang and Paschalidis [53] achieve 0.14, 0.088, 0.21, 0.082 and 0.14 F1-score on a subset of the CTU-13 dataset. In [2], 29 TCP features are extracted based on the connection period 30 s. Two feature selection algorithms are proposed, and 98.32% accuracy, 98.69% F-measure and 0.75% false positive rate are reached. Our proposed method outperforms by reaching at 99% accuracy and 99.1% F1-score. The comparison results reveal that the proposed deep learning-based approach achieves better botnet detection accuracy with very low false positive rate.

6 Conclusion

In this paper, we present a novel botnet detection by modeling network traffic traces between communication endpoints by representing the whole traffic in a graph. We extract the representative and meaningful feature set generated by flow statistics between each host for 3 different protocols, namely TCP, UDP and ICMP. We justify our approach by showing that P2P botnet and other botnets can be determined via their statistical features. For instance, the communication between each host is represented by these protocols as a matrix of summary network information. Therefore, any prior knowledge about botnet such as communication protocols (for instance, HTTP, IRC, P2P) and information about port or payload data are not required.

Then, we design the deep neural learning architecture toward botnet detection. Our particular focus is on the evaluation of the performance of the LSTM and CNN. In our evaluation study, we use two datasets publicly available for benchmarking purposes. And deep learning results are evaluated and compared with respect to the statistical metric results of baseline classifiers. The evaluation results show that the presented method outperforms other related

Table 5 Comparison of our study with other methods using the same dataset

Study and year	Method	Features	Dataset	Accuracy
Chen et al. [8], 2017	Decision tree	Conversation features (sort of flow features)	CTU-13	Accuracy: 93.6% and false positive rate: 0.3
Kirubavathi et al. [25], 2016	Naive Bayesian	4 statistical flow features	ISOT	Precision: 0.978 Recall: 0.961 F-score: 0.969 Accuracy: 99.14
Wang et al. [53], 2016	Social community detection	Flow-based features	Subset of CTU-13	Recall: 0.046 Precision: 0.80 F1-score: 0.088
Torres et al. [51], 2016	Recurrent neural network	3 flow features: size, duration and periodicity	Subset of CTU-13	Accuracy: 0.970 False Positive Rate: 0.0372
Stevanovic et al. [48], 2014	Random forest	Statistical flow features	ISOT	Accuracy: 95.7%
Alauthaman et al. [2], 2016	A classification and regression tree (entropy impurity)	Statistical flow features	ISOT and ISCX	Accuracy: 98.32%, F-measure: 98.69% and FPR: 0.75% (subject to the same set of TCP features)
Our study, 2017	Deep neural network	Statistical flow features	ISOT and CTU-13	Accuracy for ISOT, CTU-13: 0.992, 0.990, and F-score for ISOT, CTU-13: 0.973, 0.991

approaches and it can discover botnets reaching at 99% level in accuracy. We assess the run time performance of our approach. According to the run time performance test, the method costs only a small amount of time to process network flows and identifies their categories whether botnet or normal traffic. Our presented method presents acceptable training period subject to a fairly large set of botnets, very accurate detection results and the potential for real-life deployment by leveraging deep learning as an emerging solution to a large variety of classification tasks.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. ARGUS-Auditing Network Activity. https://qosient.com/argus/. Accessed: 06 Oct 2017
2. Alauthaman M, Aslam N, Zhang L, Alasem R, Hossain MA (2016) A P2P botnet detection scheme based on decision tree and adaptive multilayer neural networks. Neural Comput Appl 29:991–1004
3. Alejandre FV, Cortés NC, Anaya EA (2017) Feature selection to detect botnets using machine learning algorithms. In: International conference on electronics, communications and computers (CONIELECOMP). IEEE, pp 1–7
4. Andriesse D, Rossow C, Stone-Gross B, Plohmann D, Bos H (2013) Highly resilient peer-to-peer botnets are here: An analysis of gameover zeus. In: 2013 8th international conference on malicious and unwanted software: "the Americas" (MALWARE). IEEE, pp 116–123
5. Bou-Harb E, Debbabi M, Assi C (2017) Big data behavioral analytics meet graph theory: on effective botnet takedowns. IEEE Netw 31(1):18–26
6. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
7. Catania CA, Garino CG (2012) Automatic network intrusion detection: current techniques and open issues. Comput Electr Eng 38(5):1062–1072
8. Chen R, Niu W, Zhang X, Zhuo Z, Lv F (2017) An effective conversation-based botnet detection method. Math Probl Eng 2017:1–9
9. Chollet F et al (2015) Keras. https://github.com/fchollet/keras
10. Chowdhury S, Khanzadeh M, Akula R, Zhang F, Zhang S, Medal H, Marufuzzaman M, Bian L (2017) Botnet detection using graph-based feature clustering. J Big Data 4(1):14
11. Corp G (2017) TensorFlow: an open-source software library for machine intelligence. https://www.tensorflow.org/. Accessed 01 May 2017
12. Corp M (2017) Win32/Murlo.S. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=TrojanDownloader:Win32/Murlo.S. Accessed 06 Oct 2017
13. Corp M (2017) Win32/Sogou analysis. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Program%3AWin32%2FSogou. Accessed 06 Oct 2017
14. Cunningham P, Delany SJ (2007) k-nearest neighbour classifiers. Mult Classif Syst 34:1–17
15. Dražar M, Vizváry M, Vykopal J (2014) Similarity as a central approach to flow-based anomaly detection. Int J Netw Manag 24(4):318–336
16. Emerging threats open snort ruleset (2017) http://www.emergingthreats.net/. Accessed 15 Apr 2017
17. Garcıa S (2014) Identifying, modeling and detecting botnet behaviors in the network. Ph.D. thesis, Universidad Nacional del Centro de la Provincia de Buenos Aires
18. Garcia S, Grill M, Stiborek J, Zunino A (2014) An empirical comparison of botnet detection methods. Comput Secur 45:100–123
19. Goebel J, Holz T (2007) Rishi: identify bot contaminated hosts by IRC nickname evaluation. HotBots 7:8–8
20. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. Adaptive computation and machine learning series. The MIT Press, Boston

21. Guntuku SC, Narang P, Hota C (2013) Real-time peer-to-peer botnet detection framework based on bayesian regularized neural network. arXiv preprint arXiv:1307.7464
22. Haddadi F, Zincir-Heywood AN (2017) Botnet behaviour analysis: how would a data analytics-based system with minimum a priori information perform? Int J Netw Manag 27(4):E1977
23. iplists.firehol.org. All cybercrime IP feeds. http://iplists.firehol.org/. Accessed 15 Apr 2017
24. Jang DI, Kim M, Jung HC, Noh BN (2009) Analysis of http2p botnet: case study waledac. In: 9th Malaysia international conference on communications (MICC). IEEE, pp 409–412
25. Kirubavathi G, Anitha R (2016) Botnet detection via mining of traffic flow characteristics. Comput Electr Eng 50:91–101
26. Kudo T, Kimura T, Inoue Y, Aman H, Hirata K (2016) Behavior analysis of self-evolving botnets. In: International conference on computer, information and telecommunication systems (CITS). IEEE, pp 1–5
27. Labs F-S (2017) W32/RBot description. https://www.f-secure.com/v-descs/rbot.shtml. Accessed 06 Oct 2017
28. Labs F-S (2017) W32/Virut description. https://www.f-secure.com/v-descs/virus_w32_virut.shtml. Accessed 06 Oct 2017
29. Maaten LVD, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
30. malwaredomains.com. Malware domain blocklist. http://malwaredomains.lehigh.edu/files/domains.zip. Accessed 15 Apr 2017
31. Micro T (2017) W32/Neris description. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/worm_neeris.a. Accessed 06 Oct 2017
32. Nogueira A, Salvador P, Blessa F (2010) A botnet detection system based on neural networks. In: Fifth international conference on digital telecommunications (ICDT). IEEE, pp 57–62
33. Oujezsky V, Horvath T, Skorpil V (2017) Botnet C&C traffic and flow lifespans using survival analysis. Int J Adv Telecommun Electrotech Signals Syst 6(1):38–44
34. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830
35. Pektaş A, Acarman T (2014) A dynamic malware analyzer against virtual machine aware malicious software. Secur Commun Netw 7(12):2245–2257
36. Pektas A, Acarman T (2017) Malware classification based on api calls and behavior analysis. IET Information Security, Oct 2017. ISSN 1751-8709. URL http://digital-library.theiet.org/content/journals/10.1049/iet-ifs.2017.0430
37. Pektas A, Tankut A (2017) Effective feature selection for botnet detection based on network flow analysis. In: International conference on automatics and informatics
38. Qiu Z, Miller DJ, Kesidis G (2017) Flow based botnet detection through semi-supervised active learning. In: IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, pp 2387–2391
39. Rieck K, Schwenk G, Limmer T, Holz T, Laskov P (2010) Botzilla: detecting the phoning home of malicious software. In: Proceedings of the 2010 ACM symposium on applied computing. ACM, pp 1978–1984
40. Saad S, Traore I, Ghorbani A, Sayed B, Zhao D, Lu W, Felix J, Hakimian P (2011) Detecting p2p botnets through network behavior analysis and machine learning. In: Ninth annual international conference on privacy, security and trust (PST). IEEE, pp 174–180
41. Salvador P, Nogueira A, Franca U, Valadas R (2009) Framework for zombie detection using neural networks. In: Fourth international conference on internet monitoring and protection. IEEE, pp 14–20
42. Schmidt M, Le Roux N, Bach F (2013) Minimizing finite sums with the stochastic average gradient. Math Program 162:1–30
43. Schmock U (2000) Large deviations techniques and applications. J Am Stat Assoc 95(452):1380–1380
44. Scikit-learn: machine learning in Python. http://scikit-learn.org/stable/index.html. Accessed 15 Jan 2017
45. Sheikhan M, Jadidi Z (2014) Flow-based anomaly detection in high-speed links using modified gsa-optimized neural network. Neural Comput Appl 24(3):599–611
46. Sophos. Troj/Menti analysis. https://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/TrojMenti-A/detailed-analysis.aspx. Accessed 06 Oct 2017
47. Stevanovic M, Pedersen JM (2013) Machine learning for identifying botnet network traffic. URL http://vbn.aau.dk/ws/files/75720938/paper.pdf
48. Stevanovic M, Pedersen JM (2014) An efficient flow-based botnet detection using supervised machine learning. In: International conference on computing, networking and communications (ICNC). IEEE, pp 797–801
49. Tenebro G (2017) W32.Waledac threat analysis. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/W32_Waledac.pdf. Accessed 06 Oct 2017
50. ThreatExpert. Win32.NSIS.ay report. http://www.threatexpert.com/report.aspx?md5=eaf85db9898d3c9101fd5fcfa4ac80e4. Accessed 06 Oct 2017
51. Torres P, Catania C, Garcia S, Garino CG (2016) An analysis of recurrent neural networks for botnet detection behavior. In: Biennial congress of Argentina (ARGENCON). IEEE, pp 1–6
52. Van Der Maaten L (2014) Accelerating t-sne using tree-based algorithms. J Mach Learn Res 15(1):3221–3245
53. Wang J, Paschalidis IC (2016) Botnet detection based on anomaly and community detection. IEEE Trans Control Netw Syst 4:392–404
54. Wu T-F, Lin C-J, Weng RC (2004) Probability estimates for multi-class classification by pairwise coupling. J Mach Learn Res 5:975–1005
55. Yu H-F, Huang F-L, Lin C-J (2011) Dual coordinate descent methods for logistic regression and maximum entropy models. Mach Learn 85(1–2):41–75