Soft Computing
https://doi.org/10.1007/s00500-020-04963-z

METHODOLOGIES AND APPLICATION

DeepBot: a time-based botnet detection with deep learning

Wan-Chen Shi[1] · Hung-Min Sun[1]

Abstract
Over the decades, as the technology of Internet thrives rapidly, more and more kinds of cyber-attacks are blasting out around the world. Among them, botnet is one of the most noxious attacks which has always been challenging to overcome. The difficulties of botnet detection stem from the various forms of attack since the viruses keep evolving to avoid themselves from being found. Rule-based botnet detection has its shortcoming of detecting dynamically changing features. On the other hand, the more the Internet functionalities are developed, the severer the impacts botnets may cause. In recent years, many network devices have suffered from botnet attacks as the Internet of things technology prospers, which caused great damage in many industries. Consequently, botnet detection has always been a critical issue in computer security field. In this paper, we introduce a method to detect potential botnets by inspecting the behaviors of network traffics from network packets. In the beginning, we sample the given packets by a period of time and extract the behavioral features from a series of packets. By analyzing these features with proposed deep learning models, we can detect the threat of botnets and classify them into different categories.

Keywords Botnet · Deep learning · RNN

1 Introduction

By the end of the year 2018, according to NCTA https://cdn.ihs.com/www/pdf/enabling-IOT.pdf, it is estimated that there are over 34 billion IoT-connected devices around the world, as shown in Fig. 1. With such large amount of devices, it would be a catastrophe if just few of devices get infected by botnet virus. In 2016, a well-known virus called Mirai https://en.wikipedia.org/wiki/Mirai_(malware) was spread among IP cameras and routers and was used to perform DDoS attack on servers, causing huge damage to many online services such as Airbnb or Twitter. From 2016 to 2018, the cryptocurrency mining fever was once all over the places. People were dedicated to mine cryptocoins, including hackers. In early 2018, some hackers created viruses like Smominru https://www.cyber.nj.gov/threat-profiles/botnet-variants/smominru to force the infected system to mine cryptocurrency for them.

To solve this critical issue, a lot of researchers have proposed many methods to detect botnets. The most naive way is to build a rule-based intrusion detection system (IDS) to constantly check whether the host is under attack by observing its behaviors or identifying the existing botnet signatures. However, in order to avoid being detected by IDS, botnet viruses evolve quickly and become harder to be identified. Rule-based methods are not able to handle these continuously changing features effectively. To overcome this deficit, many dynamic methods related to machine learning or deep learning have been proposed recent years.

These machine learning based methods, on the other hand, have some difficulties in finding appropriate features to train a classification model. The quantity and the quality of features are highly related to the accuracy of the models with respect to the given datasets. Consequently, feature extraction and model selection become the major topics in botnet detection.

Communicated by V. Loia.

✉ Hung-Min Sun
hmsun@cs.nthu.edu.tw

Wan-Chen Shi
shi359@gmail.com

[1] Department of Computer Science, National Tsing Hua University, No. 101, Section 2, Kuang-Fu Road, Hsinchu 30013, Taiwan, ROC

Published online: 11 May 2020

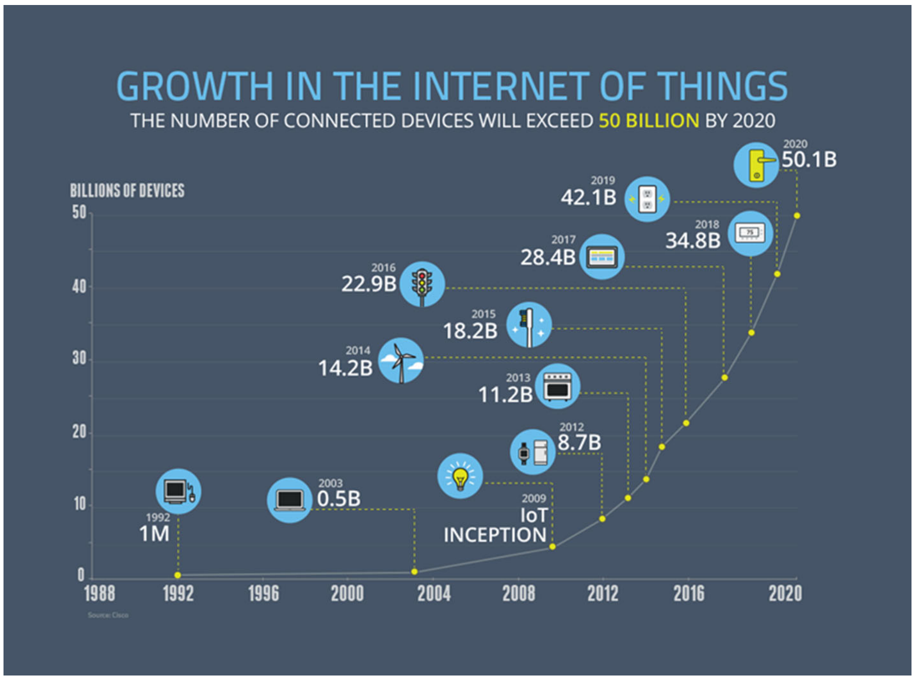


Fig. 1 The forecast of IoT devices growth

1.1 Motivation

For the past few years, many deep learning models are being applied to high-dimensional data analysis. At the same time, the rising of botnet is starting to wreak havoc to our network environment. With ease of accessibility, not only can anyone launch a DDoS attack, but also with the large amount of infected computers and IoT devices, the attack has also become much destructive. However, with their dynamically evolving features, tradition rule-based botnet detection has met their bottleneck. As the result, recent studies put emphasis on developing flexible approaches to botnet detection and classification, which demand a large amount of datasets and complex features. Therefore, deep learning seems to be a good option to tackle this problem. In this research, we aim to detect botnets with the timely variations of network traffics analytics. Among deep learning algorithms, recurrent neural networks (RNN) (Jain and Medsker 1999) and long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) are more capable of dealing with sequential data, which we believe may have higher performance than other machine learning or deep learning algorithms when it comes to analyzing context-sensitive features. Consequently, we choose to apply RNN and LSTM on time-based analysis to detect potential botnets in this paper.

1.2 Contribution

In this thesis, we propose a methodology that can analyze a series of network packet and categorize botnets with deep learning technique. The features extracted from the packet consist of information from different protocols to deal with variety of botnet structures. We use LSTM, RNN and the combination of LSTM and RNN to build botnet detection models and finally point out that the combination of LSTM and RNN is more effective in botnet classification.

1.3 Organization

The rest of this paper is organized as follows: Section 2 presents the background knowledge of botnet, RNN and LSTM. Section 3 introduces some related works in botnet detection. Section 4 describes our proposed approach and system architecture, and Sect. 5 presents our implementation in detail. Section 6 gives the evaluation of our work. In the end, all conclusions and future work are summarized in Sects. 7 and 8.

2 Background

In this section, we firstly introduce the basic concept of botnet and its structure. Secondly, we cover the background knowledge of the deep learning algorithms we use in this paper.

2.1 Botnet

A botnet consists of many network-connected devices and usually has a command and control (C&C) server which is in charge of sending commands to the rest of the devices. Generally, botnets are often used to perform distributed denial-of-service (DDoS) attacks where perpetrators can

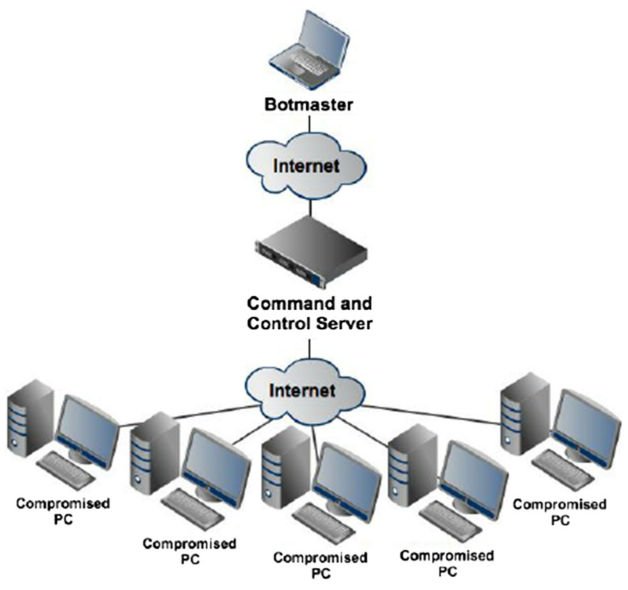


Fig. 2 Client–server botnet topology https://www.researchgate.net/figure/Typical-Client-Server-Botnet-Command-and-Control-Topology_fig1_266209917

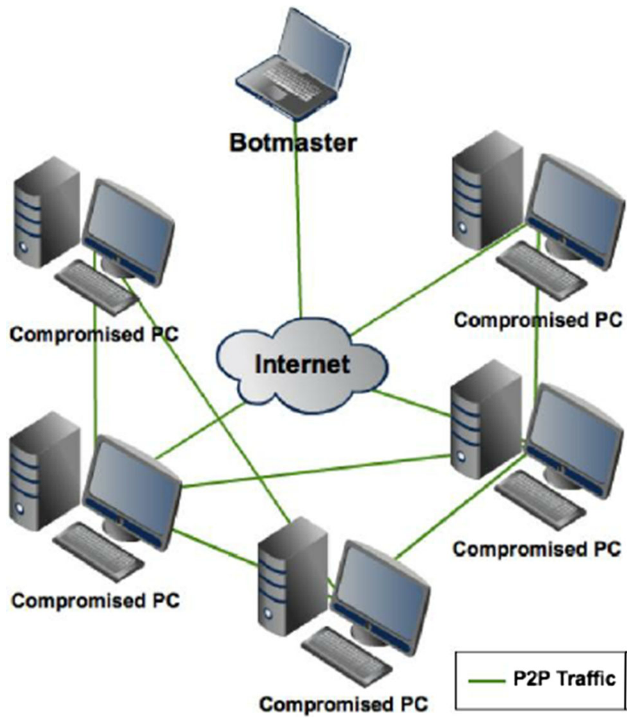


Fig. 3 P2P botnet topology https://www.researchgate.net/figure/Typical-Client-Server-Botnet-Command-and-Control-Topology_fig1_266209917

attack their victims with thousands of devices. Additionally, botnets can also be used to deliver spam since sending spam needs many mail servers as most of them get banned from service providers. In the next section, we describe more details about the architecture of botnets.

Mainly, botnet architectures can be categorized into two types by the way bots communicate: client–server model and peer-to-peer (P2P) model. The client–server model is shown in Fig. 2. Client–server model is a typical structure of botnet. It is composed of a botmaster, several C&C servers and a lot of bots (or botclients) which are infected by the botnet viruses. A botmaster is like a commander; it controls the whole botnet remotely by sending commands to a C&C server. A C&C server, on the other hand, passes the commands from a botmaster to the bots. Additionally, a botmaster can learn the number of botclients and their information from the C&C servers in a botnet. The rest of the bots, the compromised devices, execute the given commands stealthily in the mean time. This traditional kind of botnet, however, can be easily discovered since we can track the hostname or the IP address of C&C servers and add them to a blacklist. To avoid from revealing themselves, some botnets render a large number of hostnames dynamically by domain generation algorithm (DGA) https://en.wikipedia.org/wiki/Domain_generation_algorithm.

Another approach to prevent from being detected is to construct a botnet with P2P model. As we can see from Fig. 3, instead of communicating with a centralized server, a bot can act as a commander and a receiver at the same time. Firstly, to find out its neighbors in the botnet, a bot keeps probing random IP addresses until it finds out other infected devices. Then, when a bot receives a command, it executes the command and distributes the command to other bots. In this way, it would be hard to find suspicious IP addresses and track down the source of the botnet compared to centralized botnet architecture.

2.2 Neural network

Artificial neural network is now widely applied to many fields. It simulates the properties of biological neural network to solve cognitive problems like image recognition. Many deep learning algorithms like convolutional neural network (CNN) or RNN are built on the basis of neural network.

2.3 Neural network structure

A neural network consists of three different kinds of layers: input layer, hidden layer and output layer. A neural network usually has several hidden layers for data processing. Each hidden layer contains more than one neuron, which are used to extract certain features from the input. Figure 4 illustrates the architecture of an individual neuron.

Typically, a neuron has two parts: The first part sums up the input data according to their weights, and the other part, also called activation function, transforms the result into nonlinear

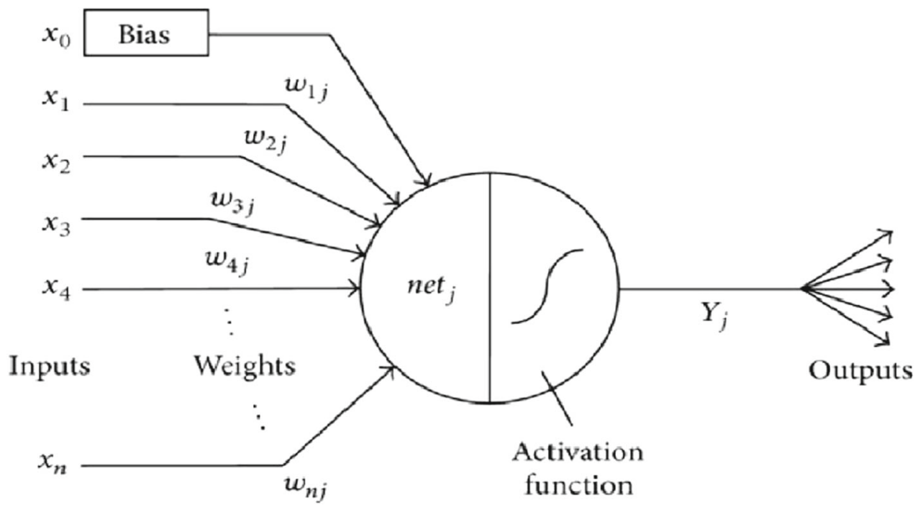


Fig. 4 Architecture of an individual neuron

form. After performing the above operations in hidden layers, we do backpropagation for optimization and then flatten the data into one dimension to get our final result from output layer.

2.4 Activation function

A neural network aims to learn complicated functional mappings from data, so it is preferred to be a nonlinear model rather than a linear one. The objective of an activation function is to transform the aggregated results into nonlinear form. Normally, we define a threshold in an activation function. If the weighted sum of a given node is larger than the threshold, it is "activated"; otherwise, it is "deactivated." The outputs from the activated nodes are then fed into next layer.

2.5 Backpropagation

After processing input data layers by layers, we do backward propagation to minimize the cost by optimizing the parameters in neural network. The work flow is illustrated in Fig. 5. For each layer, we calculate the partial derivative of cost $J(\theta)$ with respect to each parameter θ and propagate the error backward. In each propagation, the parameters are adjusted to minimize the cost. After performing above computations repeatedly until the cost converges, we can find the optimal set of parameters.

2.6 RNN

Among all the deep learning algorithms , RNN is more capable of dealing with sequential data like time series, audio or video due to its characteristic. Traditional neural networks conceive each input as separate data, so all the inputs and outputs are independent of each other. However, for sequential data, each chunk of input is highly related to its previous

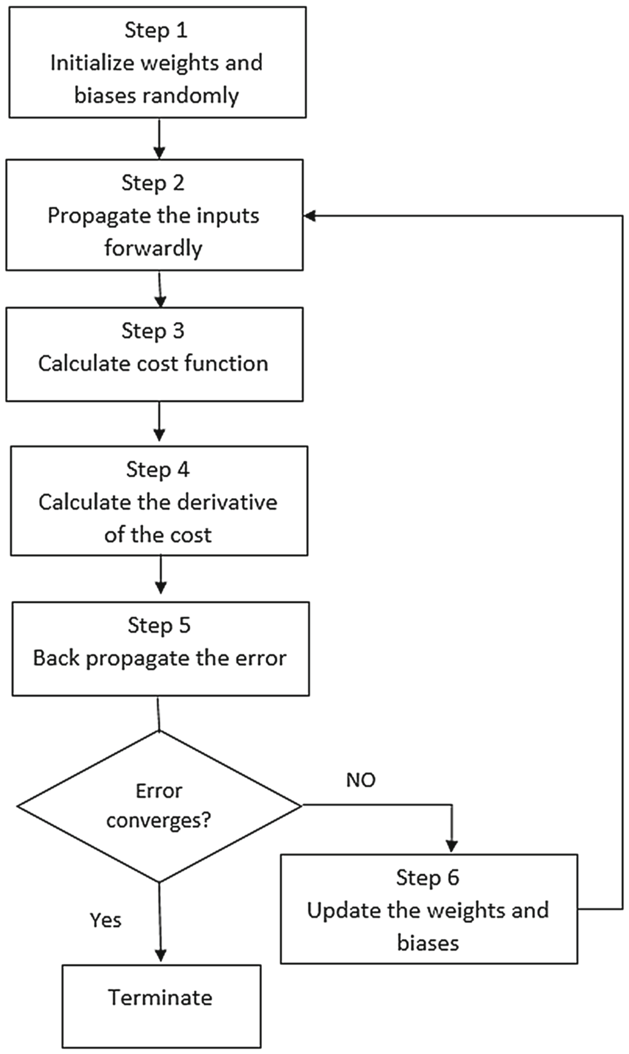

Step 1
Initialize weights and biases randomly
Step 2
Propagate the inputs forwardly
Step 3
Calculate cost function
Step 4
Calculate the derivative of the cost
Step 5
Back propagate the error
Error converges?
NO
Step 6
Update the weights and biases
Yes
Terminate

Fig. 5 Backpropagation process

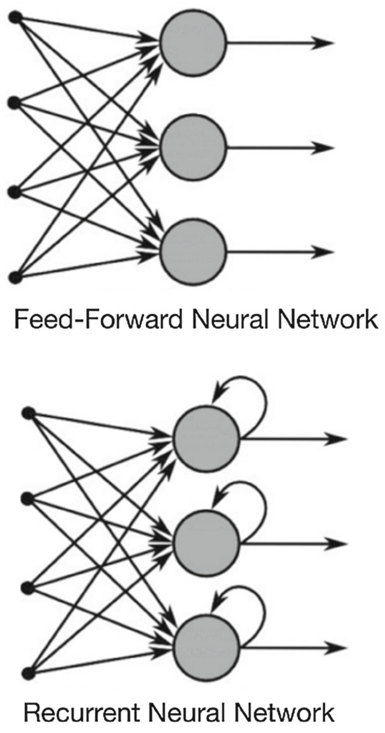


Fig. 6 Comparison of feed-forward and recurrent network

and next input; hence, we need extra memory to keep track of a series of data.

As we can see from Fig. 6, in traditional neural network, we use feed-forward method to generate outputs, which means the data flow is one directional, so each input can only be processed once and never be fed into the same neural network again. In RNN, on the contrary, the input can be stored temporarily and taken into consideration when we process next batch of input. Therefore, the general summation function of a neuron can be defined in Eq. 1:

$$h_t = f(W_h h_{t-1} + W_x x_t) \tag{1}$$

where W is the weight of given state, h_{t-1} denotes the state at time $t-1$, and x_t is the input at time t.

Furthermore, unlike in CNN or other algorithms, tanh is chosen as activation function in RNN rather than sigmoid or ReLU. The tanh function is shown in Eq. 2:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

2.7 LSTM

The shortcoming of RNN, however, is vanishing gradient problem, discovered by Hochreiter (1998). During backpropagation, the gradients of loss are apt to get smaller and might be closed to zero as we move backward. To solve this problem, LSTM introduces memory block in the neural network. The structure of memory cell in LSTM is shown in Fig. 7.

There are three gates in a memory block: forget gate, input gate and output gate. Firstly, the sigmoid function in forget gate decides whether the information should be used or discarded. Secondly, in input gate, a sigmoid function decides which values should be updated, and then, a tanh function creates a vector of parameters for the selected values. In output gate, a sigmoid function selects which parts of the cell state should be output. Finally, the cell state is computed by a tanh function and multiplied by the output of a sigmoid function. The above steps are done in each hidden layer.

2.8 Applications of RNN and LSTM

As mentioned above, because of its structure, RNN and LSTM are capable of memorizing information of past states, making them suitable for analyzing sequential data. For recent year, RNN have achieved great success in the category of natural language processing (NLP), such as language modeling (Mikolov et al. 2010), generating text (Sutskever et al. 2011) and text translation (Sutskever et al. 2014). In addition, if working with convolutional neural networks (CNN), RNN and LSTM can be used in image captioning, which means this combined model can generate text that describes the image. What is more amazing is that, when visualizing the result, the researcher found out the generated words even align with features found in the images (Xu et al. 2015).

3 Related work

Botnet detection has always been a critical issue. Many approaches have been proposed and can be roughly categorized into four types: signature-based, anomaly-based, DNS-based and mining-based (Feily et al. 2009). In this section, we discuss some techniques according to the categories mentioned above.

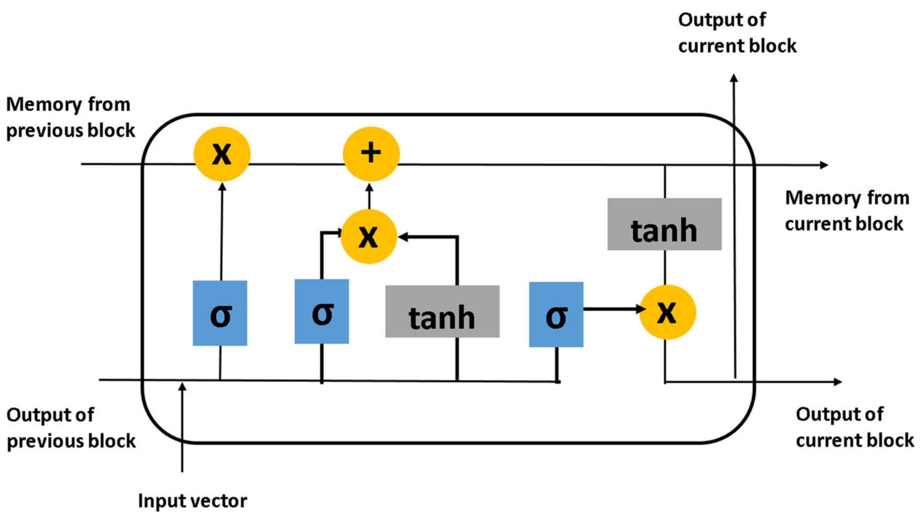
3.1 Signature-based

Signature-based method detects botnets with the signatures of currently known botnets. It was once popular and was adopted by some IDS systems like Snort https://www.snort.org/. However, it can be used only under the circumstances that botnets are known to the system, which is not feasible nowadays since botnets are mutating too fast for IDS to update.

3.2 Anomaly-based

Anomaly-based approach monitors system activities and network traffics. IDS can detect botnets in time when the

Fig. 7 LSTM structure



status becomes abnormal, even they are unknown to the system. Siboni and Cohen (2014) utilized Lempel–Ziv universal compression algorithm, LZ78 (Ziv and Lempel 1978), to estimate the likelihood of network traffics. Binkley and Singh (2006) presented an algorithm that combines TCP-based anomaly detection with IRC statistics to detect botnets. Nonetheless, this approach can be defeated if the IRC commands are encoded with a trivial cipher. Karasaridis et al. (2007) used passive analysis based on flow data in transport layer to detect and characterize botnets. Their algorithm can detect encrypted communications in botnets, solving the problem in Binkley and Singh (2006).

3.3 DNS-based

DNS-based method is similar to anomaly-based detection. It collects DNS information in botnets and monitors DNS traffics as botnets need to rally infected hosts by sending DNS queries. Additionally, in order to hide their identities, botmasters apply DGA to update their domain names dynamically. Villamarin-Salomon and Brustoloni (2008) identified C&C servers by analyzing recurring dynamic DNS (DDNS) replies with nonexistence domains (NXDOMAIN). Also, Choi et al. (2007) introduced a detection method by monitoring group activities in a local network.

3.4 Mining-based

Mining-based techniques include machine learning or deep learning methods, which are widely used recent years. Many researchers combine traditional ways with machine learning algorithms to achieve better performance. Gu et al. (2008) developed a framework, *BotMiner*, to detect botnet by using unsupervised learning. The work in *BotMiner* could be divided into two parts, both of which are run in parallel. On the one hand, they analyzed the likelihood of communication patterns among groups of hosts. On the other hand, they analyzed the anomaly payloads in network packets. However, Gu et al. (2008) pointed out that adversaries still can evade from the detection if the botnets act more randomly. Homayoun et al. (2018) used autoencoder and CNN to detect botnet traffic and proved that autoencoder performed better than CNN. Homayoun et al. (2018) also suggested that LSTM can be a future work. Tran et al. (2018) presented a novel LSTM approach to do DGA botnet detection. The approach combined binary and multiclass classification models to solve multiclass imbalance problem in LSTM. Vinayakumar et al. (2019) also tackle with DGA botnet detection. They use convolutional neural network with a long short-term memory (CNN-LSTM) pipeline as model. Their result shows they can achieve high accuracy and low false positive rates. At the same time, since their architectures are quiet simple, they have shorter training time compared to other DGA botnet detection framework. In Wang et al. (2017), the authors proposed a technique that does not require hand-designed features but raw traffic as input data. Instead, Wang et al. (2017) converted different botnet traffics into images and classified these images with CNN.

4 Methodology

In this section, we describe the architecture of our design. Our methodology can be divided into four sections as below:

- Overall workflow
- Dataset
- Feature selection
- Model

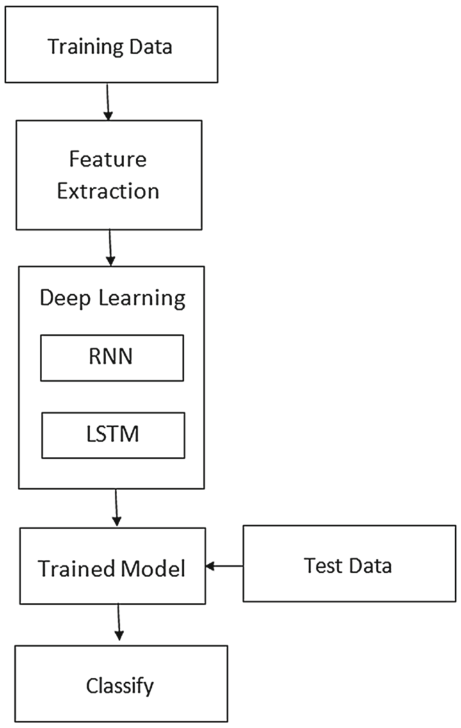


Fig. 8 System workflow

4.1 Workflow

Figure 8 depicts our framework design. In the first step, network traffic is recorded and stored into pcap format. In step two, we extract features from the input network traffics we get from step one. Then, in step three, we use three different methods to train our model during training. After the model is built, we can input test data to the model and get the classification result.

4.2 Dataset

In this paper, we aim at detecting bots that are infected by malware. We train our model with datasets from malware capture facility project (MCFP) (García et al. 2014), which is conducted by researchers in Czech Technical University. MCFP has the largest botnet datasets so far, and the dataset is continually updated. In this project, the authors captured network traffics on infected virtual machines and try to collect communication pattern between infected bots and C&C servers.

Among different kinds of botnet families, we select four kinds of botnet: Dridex, Emotet, Sality and Zbot, due to the fact that their datasets are larger, plus these families have caused huge impacts for several years.

4.3 Feature extraction

To train our model, we extract 35 network traffic features from input data. Packets are sampled every 10 seconds, and the features are hence derived from these samples. After feature extraction, the results are then saved into csv file format. Below, we categorize these features and explain them in detail. All the features are listed in Table 1.

Table 1 Feature and description

Feature	Description
AvgTimeDelta	Average time delta of captured packets
AvgPktSz	Average packet size
AvgBytePS	Average bytes in one second
PktNum	Number of packets transmitted
PktPS	Number of packets in one second
SameSzRatio	Ratio of packets having same size
InPktRatio	Ratio of packets received
TotalByteIn	Total bytes received
AvgTCPSgmtLen	Average length of TCP segment
PktOutOfOrder	Number of out-of-order packets
PktACKDup	Number of duplicate ACK packets
TCPRatio	Ratio of TCP packets
ConvNum	Number of conversations
AvgConvDelta	Average time delta in a conversation
AvgNxtConvDelta	Average time delta between two conversations
AvgConvPktNum	Average packet numbers in a conversation
AvgConvDur	Average duration of conversations
SYNRatio	Ratio of SYN packets
PSHRatio	Ratio of PSH packets
FINRatio	Ratio of FIN packets
RSTRatio	Ratio of RST packets
HTTPRatio	Ratio of HTTP packets
AvgHTTPQryLen	Average length of HTTP queries
AvgHTTPRspLen	Average length of HTTP responses
HTTPGetNum	Number of HTTP GET requests
HTTPPostNum	Number of HTTP POST requests
UDPRatio	Ratio of UDP packets
AvgUDPLen	Average length of UDP packets
DNSQryNum	Number of DNS queries
AvgDNSQryLen	Average length of DNS queries
AvgDNSDelta	Average time delta of DNS queries
ICMPRatio	Ratio of ICMP packets
SMTPRatio	Ratio of SMTP packets
DNSRatio	Ratio of DNS packets
IRCRatio	Ratio of IRC packets

– *Protocol features*
Botnet can be executed in many ways with different protocols, like ICMP flood or mail spam, which is executed via SMTP protocol. In order to detect abnormal behavior, we need to monitor the ratio of each protocol listed below:

1. UDP ratio
2. TCP ratio
3. HTTP ratio
4. ICMP ratio
5. SMTP ratio
6. IRC ratio
7. DNS ratio

– *TCP flag*
TCP flags denote status of connections. They are frequently used in three-way handshake. Thus, it is possible to find out the communication patterns between bots and C&C server by inspecting TCP flags. Additionally, some attacks like SYN flood or RST attack utilize these flags to interfere victims' connection:

1. SYN ratio
2. PSH ratio
3. FIN ratio
4. RST ratio

– *HTTP features*
HTTP is commonly used by many services or applications; therefore, hiding botnet flows in HTTP traffics can prevent it from being detected. Compromised bots may continually send requests to web servers:

1. average length of HTTP queries
2. average length of HTTP responses
3. number of GET requests
4. number of POST requests

– *Conversation features*
When bots communicate with C&C servers, the length and the duration of their conversations are apt to be similar. Also, studying the frequency of the conversations helps us to discover their communication pattern:

1. Number of conversations in a time period
2. Average time delta between two conversations
3. Average time delta of packets in a conversation
4. Average packet numbers in a conversation
5. Average duration of conversations

– *Other features*
The rest of the features are derived from the size and time duration of a series of packets. Analyzing these characteristic of a bunch of packets can be useful to differentiate various kinds of botnets. For example, the size of packets is tend to be the same, or the number of packets in a short period of time may be different during a attack. These features are given in Table 1

4.4 Models

We propose three different models to train our datasets: RNN, LSTM and the combination of RNN and LSTM, so-called combinational model. In the combinational model, we combine LSTM layers and RNN layers together.

5 Implementation

5.1 Tools

To extract features from pcap files, we use the following tools:

1. Python 3.6
2. Python library: PyShark https://kiminewt.github.io/pyshark/

PyShark is a powerful tool for network packet analysis. It is a Python wrapper of tshark https://www.wireshark.org/docs/man-pages/tshark.html, which is a lightweight library in comparison with other libraries like scapy or pypcapfile since it simply leverages functionalities in tshark.

Our training models are implemented with the following libraries in Python:

1. Keras v2.2.4
2. Numpy v1.16.1
3. Pandas v0.24.1
4. Scikit-learn v0.20.2
5. Tensorflow v1.6.0

We use Pandas and Numpy to preprocess extracted features from input data before building training models. During training, Keras and scikit-learn are used to build our models, while Tensorflow is run as backend in Keras. Figure 9 shows all the tools we use.

The experiment is run on Windows 10 enterprise with hardware settings as follows: Intel i5-8400K @ 4GHz CPU, 32-GB RAM, Geforce GTX 1070Ti.

5.2 Dataset

There are four botnet families selected from MCFP datasets: Dridex, Emotet, Sality and Zbot, as we previously described in Sect. 4.2. The network traffic size of each family is 1.33 GB, 580 MB, 3.08 GB, 1.88 GB. Moreover, to distinguish normal network traffics from abnormal traffics, we collect 15 GB network traffics from normal users.

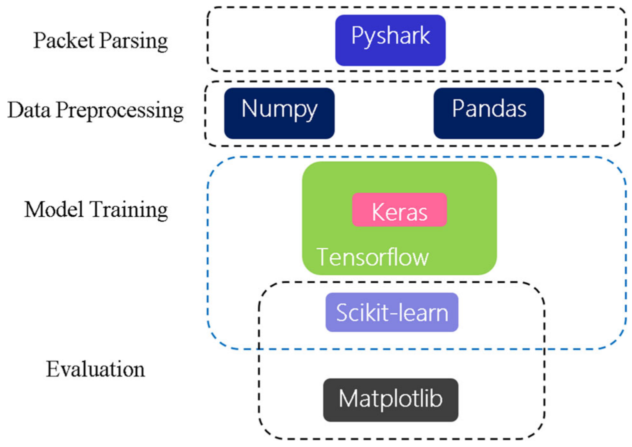


Fig. 9 Implementation tools

5.3 Packet parsing

We read each packet from a input file with pcap format. There are three steps in packet parsing:

- *Step 1* Calculate the size of each packet, time delay between previous packet, time delay between conversations and duration of a conversation.
- *Step 2* Inspect protocols in each layer, from transport layer to application layer. Then, we extract the desired information according to protocol types.
- *Step 3* Aggregate the results of previous two steps and output to csv format. The aggregation process is shown below:

5.4 Deep learning

To train our models, we firstly split every 600 rows as one data point. In a data point, each row of data is timely dependent to its previous and next rows. Thus, the sequence in a data point will not be shuffled during the training.

Secondly, we do data scaling by standardizing the input. Without standardization or normalization, the training process will become slow and unstable, which may result in convergence failure.

Next, we shuffle the data and split them into training set and test set with split ratio 0.8. The purpose of data shuffling is to prevent from overfitting since the order of data will influence training process.

Then, we feed the training data into three different models: LSTM, RNN and combinational model.

During training process, each hidden layer is followed by a dropout layer. The purpose of dropout is to randomly discard some of the output to avoid overfitting and accelerate the training process. Figure 10 depicts the idea of dropout layer. This concept is firstly proposed by Srivastava et al. (2014).

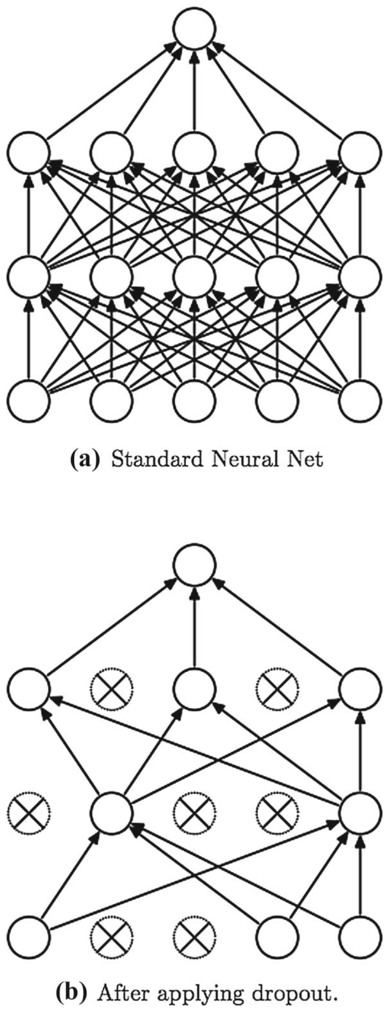


Fig. 10 Dropout layer

At the end of each epoch, we do cross-validation with 30% of the training set to ensure the models are not overfitting.

At the output layer of our models, we choose softmax https://en.wikipedia.org/wiki/Softmax_function as our activation function to categorize the final output.

6 Evaluation

6.1 Evaluation method

To measure the performance of these models, we compare the results with accuracy, precision, recall and F1 score. Solely, comparing accuracy rate is not enough to evaluate the performance of a model since we only evaluate the percentage of correctly classified samples when calculating accuracy rate, but we ignore the samples that are classified incorrectly. To cope with it, we need to interpret the results from different aspects in different situations.

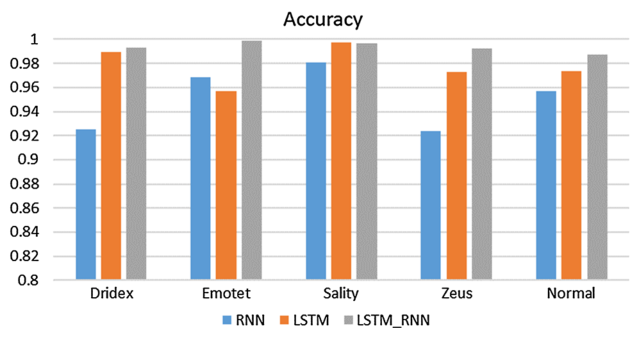


Fig. 11 Accuracy of each class in each model

The evaluation formulae are defined below:

$$\text{Accuracy} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}} \quad (3)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (5)$$

$$F1\ \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6)$$

Equation 4 measures the fraction of botnet families that were correctly classified as a particular class among all test samples that actually belong to that class, while Eq. 5 is a measure of the fraction of botnet families that were correctly classified as a certain class among all test samples that were predicted as that class.

6.2 Comparison

Firstly, we compare the accuracy of each model, as shown in Fig. 11. Generally speaking, LSTM and the combinational model have higher accuracy than RNN.

From Fig. 12, we can see that the recall rate of Zeus traffics is lower than other traffics in LSTM model, indicating that LSTM may incorrectly predict Zeus to other classes.

Figure 13 shows that the precision rates in RNN model are the lowest in average. Also, the precision rates of Emotet traffics and normal traffics are relatively low in LSTM model, which implies that some of botnet families are mistaken for Emotet and normal traffics by LSTM model (Fig. 14).

On the other hand, the precision rate of normal traffics in the combinational model is higher than those in the other models. We can therefore conclude that most of the time, it does not classify botnet traffics to normal traffics, which makes it a preferable choice for botnet detection.

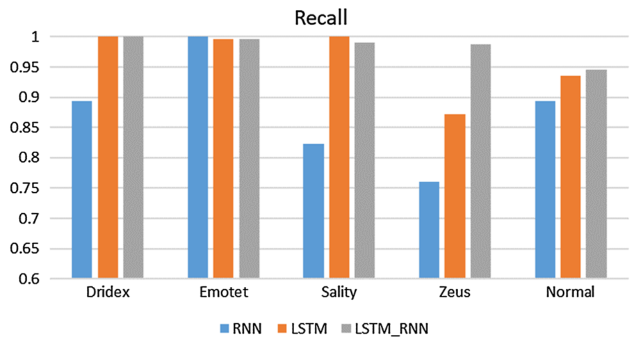


Fig. 12 Recall of each class in each model

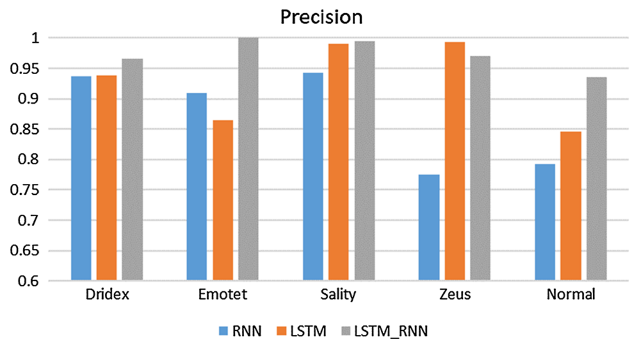


Fig. 13 Precision of each class in each model

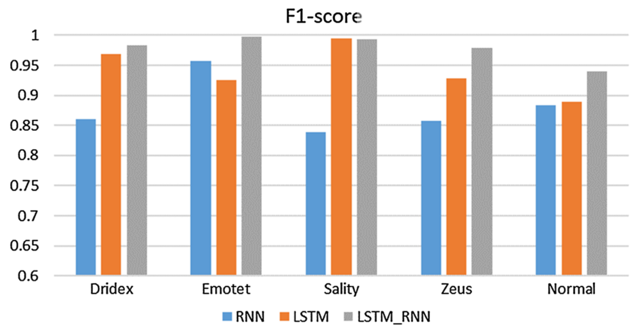


Fig. 14 F1 score of each class in each model

Nevertheless, in real-world circumstances, botnet traffics do not appear as frequently as normal traffics do. To simulate real-world situation, we test our models with 90% of normal traffics and 10% of botnet traffics. The result is shown in Table 2, and the overall performance of these three models is listed in Table 3.

In conclusion, if we compare these three models, we can find out that the combinational model has the best performance since its average F1 score is the highest.

7 Conclusion

In this paper, we propose a method to identify and classify botnet by analyzing time-based features of network traffics.

Table 2 Performance of simulated real-world circumstances

	Accuracy	Precision (%)	Recall (%)	F1 score (%)
RNN	92.02	94.68	90.49	92.52
LSTM	99.69%	99.83	90.49	94.82
LSTM_RNN	92.33%	97.67	88.34	92.65

Table 3 Overall performance

	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
RNN	95.06	88.12	88.92	88.52
LSTM	97.78	92.77	96.59	94.64
LSTM_RNN	99.36	97.97	98.86	98.42

The feature sets do not exclusively aim at certain type of botnet; instead, it contains several kinds of features from different protocols. As a result, not only can it detect different kinds of major botnets, but also able to adjust itself to the situation when those botnet changes the way they communicate or attack. The proposed method can be implemented on IDS so that users can monitor network traffics and feed the recorded packets into classification model to detect potential botnet traffic.

Moreover, we compare our innovative model (hybrid of LSTM and RNN) with the other two traditional models (LSTM and RNN) under four common metrics (accuracy, precision, recall and F1 score). In general, LSTM surpasses RNN in these metrics. However, with the aid of RNN, the LSTM can achieve better result. As a consequence, we finally point out that the combination of LSTM and RNN can be a powerful model to detect botnets.

8 Future work

Botnet identification normally shows detail of C&C servers or malicious connections. This work, however, does not provide much information about botnet. Our future work will thus be directed at collecting more details about C&C servers and other information.

In addition, the way botnet acts might change over time, so it is important to keep our model up to date. Maybe some online learning technique can make us respond more quickly, so we do not have to train the whole model again with new data.

LSTM does solve vanishing gradient problem that RNN suffers. Combining these two algorithms is also proved to be more efficient in this paper. However, RNN and LSTM consume lots of memory resource during training in order to memorize previous status, which is not hardware friendly and thus may take more time to train.

A recent study (Bai et al. 2018) shows that temporal convolutional network (TCN) outperforms LSTM and can process longer effective memory with less resource. In addition, TCN does not have exploding or vanishing gradients problems due to its architecture. Therefore, detecting botnet with TCN can be considered as our future work.

Compliance with ethical standards

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

References

Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. CoRR, vol abs/1803.01271 [Online]. arXiv:1803.01271

Behind the numbers: growth in the Internet of Things (2015). https://cdn.ihs.com/www/pdf/enabling-IOT.pdf

Binkley JR, Singh S (2006) An algorithm for anomaly-based botnet detection. In: Proceedings of the 2nd conference on steps to reducing unwanted traffic on the internet. USENIX Association [Online]. http://dl.acm.org/citation.cfm?id=1251296.1251303

Botnet topology. https://www.researchgate.net/figure/Typical-Client-Server-Botnet-Command-and-Control-Topology_fig1_266209917

Choi H, Lee H, Lee H, Kim H (Oct 2007) Botnet detection by monitoring group activities in DNS traffic. In: 7th IEEE international conference on computer and information technology

Dynamic generation algorithms. https://en.wikipedia.org/wiki/Domain_generation_algorithm

Feily M, Shahrestani A, Ramadass S (2009) A survey of botnet and botnet detection. In: Third international conference on emerging security information, systems and technologies

García S, Grill M, Stiborek J, Zunino A (2014) An empirical comparison of botnet detection methods. Comput Secur 45:100–123. https://doi.org/10.1016/j.cose.2014.05.011

Gu G, Perdisci R, Zhang J, Lee W (2008) Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Proceedings of the 17th conference on security symposium, ser. SS'08. USENIX Association [Online]. http://dl.acm.org/citation.cfm?id=1496711.1496721

Hochreiter S (1998) The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int J Uncertain Fuzziness Knowl-Based Syst. https://doi.org/10.1142/S0218488598000094

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput. https://doi.org/10.1162/neco.1997.9.8.1735

Homayoun S, Ahmadzadeh M, Hashemi S, Dehghantanha A, Khayami R (2018) BoTShark: a deep learning approach for botnet traffic detection. Springer, Cham, pp 137–153

Jain LC, Medsker LR (1999) Recurrent neural networks: design and applications, 1st edn. CRC Press Inc, Boca Raton

Karasaridis A, Rexroad B, Hoeflin D (2007) Wide-scale botnet detection and characterization. In: Proceedings of the first conference on first workshop on hot topics in understanding botnets. USENIX Association [Online]. http://dl.acm.org/citation.cfm?id=1323128.1323135

Mikolov T, Karafiat M, Burget L, Cernocky J, Khudanpur S (2010) Recurrent neural network based language model. In: International speech communication association, pp 1045–1048

Pysahrk. https://kiminewt.github.io/pyshark/

Siboni S, Cohen A (2014) Botnet identification via universal anomaly detection. In: 2014 IEEE international workshop on information forensics and security (WIFS), pp 101–106

Smominru (2018). https://www.cyber.nj.gov/threat-profiles/botnet-variants/smominru

Snort (2016). https://www.snort.org/

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958. http://dl.acm.org/citation.cfm?id=2627435.2670313

Sutskever I, Martens J, Hinton G (2011)Generating text with recurrent neural networks. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp 1017–1024

Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in neural information processing systems 27(NIPS 2014), pp 3104–1112

Tran D, Mac H, Tong VT, Tran HA, Nguyen LG (2018) A LSTM based framework for handling multiclass imbalance in dga botnet detection. Neurocomputing 275:2401–2413

Tshark. https://www.wireshark.org/docs/man-pages/tshark.html

Villamarin-Salomon R, Brustoloni JC (2008) Identifying botnets using anomaly detection techniques applied to DNS traffic. In: 2008 5th IEEE consumer communications and networking conference, pp 476–481

Vinayakumar R, Soman K, Poornachandran P, Alazab M, Jolfaei A (2019) DBD: deep learning dga-based botnet detection. In: Deep learning applications for cyber security. Springer, Cham, Switzerland, 2019, pp 127–149

Wang W, Zhu M, Zeng X, Ye X, Shengand Y (2017) Malware traffic classification using convolutional neural network for representation learning. In: 2017 International conference on information networking

Wikipedia: Mirai. https://en.wikipedia.org/wiki/Mirai_(malware)

Wikipedia: Softmax function. https://en.wikipedia.org/wiki/Softmax_function

Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhutdinov R, Zemel R, Bengio Y (2015) Show, attend and tell: neural image caption generation with visual attention. In: Proceedings of the 32nd international conference on machine learning, vol 37

Ziv J, Lempel A (978) Compression of individual sequences via variable-rate coding. In: 1978 IEEE transactions on information theory, pp 530–536