HTTP Tunnel Trojan Detection Model Based on Deep Learning

Yubo He¹, Yuefei Zhu¹ and Wei Lin¹

¹State Key Laboratory of Mathematical Engineering and Advanced Computing, Information Engineering University, Zhengzhou, 450001, China

Abstract: Aiming at reducing the high False Negative rate of the existing Trojan horse detection method based on behavior, this paper utilized the sequence characteristics of tunnel Trojan communication extracted from the transport layer and the bi-directional recurrent neural network in deep learning to build a HTTP tunnel Trojan detection model. The experimental result showed that the deep learning-based detection model reduced the false positive rate of normal network traffic and improved the Trojan detection rate. We also found that the deep learning-based detection model reduced the feature selecting and data cleaning process of generating samples and improved the easy-using of the HTTP tunnel Trojan detection model.

1. Introduce
As one of the key technologies in cyberspace security, trojan detection can guarantee the security of cyberspace. How to detect trojans in a timely and effective way has become a hotspot in the network security researcher’s community. Trojan detection is divided into two parts, host-based trojan detection and network-based Trojan detection. The network-based Trojan detection technology distinguishes Trojan traffic from normal traffic by identifying characteristics of network traffic behavior, traffic characteristics, and other traffic transmissions to identify Trojan traffic. In the early days, people used the signature matching method to detect the Trojan traffic, which has achieved high detection rate for the specific Trojans. However, with the appearance of Trojan variant and the new Trojan, the Trojan flow detection based on the signature matching cannot detect unknown Trojans, and Trojan detection based on signature matching may get high false positive rate, which causes false alarm; and a large amount of human and material resources must be consumed to maintain the signature databases of Trojan traffic character features to identify Trojan traffic and to identify variant Trojans. The behavior-based Trojan detection utilizes the unique communication behavior of the Trojan traffic, such as the downlink packet ratio and the heartbeat behavior[1] to identify Trojan traffic and to identify variant Trojans. The behavior-based Trojan detection utilizes the unique communication behavior of the Trojan traffic, such as the downlink packet ratio and the heartbeat behavior[1] to identify Trojan traffic, which can identify the variant Trojan and effectively improve applicability the Trojan detection.. With the continuous development of data mining algorithms, data mining algorithms can automatically calculate the behavior pattern of abnormal traffic through input characteristics. The general processing of malware traffic detection based on machine learning is to preprocess the raw traffic, extract the characteristics, input it into the data-mining algorithm and obtain a malware traffic classifier.

Recently, with the maturity of deep learning theory, the research on malware network traffic detection based on deep neural network model has yielded very significant results. Wang[2] proposed a method which converted the raw traffic directly into pictures and used self-encoding network to identify network traffic, and gained high classification accuracy; Wang Yong[3] proposed a way which converted the network traffic characteristics into pictures and put the pictures into the convolutional neural network to classify; Wang Wei[4] cut the original traffic into same length and directly put the raw traffic into the
convolutional neural network to learn the raw traffic’s character characteristics. However, most of today's deep learning algorithms focus on the character characteristics of traffic, and classifying traffic by self-learning of character features, but compared to normal traffic, Trojan traffic is a time series with very obvious timing characteristics, and Trojan traffic are usually encrypted, which is more difficult to extract the character characteristics. In order to solve these problems, this paper used the recurrent neural network to extract the timing characteristics of Trojan traffic and reduced the false positive rate of deep learning model to identify Trojan traffic.

2. Trojan communication behavior analysis

The HTTP tunnel Trojan is a Trojan that transmits control commands, user data, sensitive data and so on through an HTTP tunnel. Compared with the traditional Trojan, the HTTP tunnel Trojan establishes a connection with the server through the universal port 80. The traditional firewall directly releases the HTTP data, so the HTTP tunnel Trojan can escape the detection by the firewall, creating a secure Trojan transmission environment.

Since the difference between the HTTP tunnel Trojan and the ordinary Trojan is only the communication channel, there is a big difference between the HTTP tunnel Trojan traffic behavior and the normal traffic behavior, which is the same as the ordinary Trojan. Like the normal application, HTTP tunnel Trojan needs to establish a client-server connection and communication. The HTTP tunnel Trojan communication is usually divided into three phases.

1. Connection establishment. After the console program rooted in the Trojan Controlled Terminal runs, the Trojan sends a DNS request, establishes a TCP connection with the IP address of the controlling terminal obtained by the DNS server and transfers data through an HTTP tunnel with the controlled. While the control terminal is offline, the control terminal receives RST package.

2. Data transmission. The controlled terminal receives the control command information from the control terminal, executes the control command and sends the return result. Since the Trojan controlled terminal is similar to the role that server do in the client/server architecture, the network traffic behavior of the Trojan horse program will be different from the normal program network traffic during data transmission, that is, the Trojan has external control features, and the normal application has internal control features, as shown in Figure 1.

3. Connection maintenance. In order to maintain the network connection between the controlled terminal and the controlling terminal and probe the network status of the controlled terminal, the attacker introduces a heartbeat mechanism while designing the Trojan. The controlled terminal program will send a data packet containing the host specific information to the controlling terminal through the network connection. The controlling terminal can complete the handshake with the controlled terminal and confirm the network status of the controlled terminal.

As shown in Figure 1, the Trojan has external control features while the normal application has
internal control features, therefore during data transmission, the Trojan traffic often begins with the first downstream data packet while the normal application starts with the first upstream data packet. So we can assume that after capturing raw network traffic in the local host and filtering heartbeat packet, if the data transmission behavior is similar to Figure 1(a), it’s considered to be a normal program’s traffic; if it is similar to Figure 1(b), then it’s considered to be a trojan’s traffic.

3. HTTP tunnel trojan detection model based on deep learning

In order to verify the validity of the Trojan detection model proposed in this paper, the structure of the Trojan detection model is shown in Figure 2. The raw traffic is preprocessed to obtain the network traffic characteristics, after pre-processing, the network traffic characteristics is put into the long short-term network to learn the behavior feature and constructs a Trojan traffic classifier.

![Figure 2. Architecture of HTTP tunnel trojan detection model](image)

3.1. Network traffic capture module

The network traffic capture module is mainly developed by winpcap. The winpcap module captures raw network traffic and saves it in pcap file, and winpcap module can parse and extract communication information such as transmission protocol and communication payload of network data packet. In this experiment, the traffic data of the network was mainly taken from the exit of a laboratory LAN, and the network traffic of the LAN was captured by the mirror port of the switch.

3.2. Data preprocessing

The preprocessing of network traffic mainly includes two modules, traffic clustering module and heartbeat filtering module. The traffic clustering module splits the captured traffic data packets into clusters according to an algorithm; the heartbeat filtering module uses the association rule algorithm to filter the heartbeat packet in each data packet cluster, which gets the pure data transmission traffic. The Data prepossessing process is shown in Figure 3.

![Figure 3. Data Prepossessing process](image)
3.2.1. Traffic slicing module. HTTP traffic has self-correlation and self-similarity properties. But HTTP traffic will appear self-similarity over a long time-span, and HTTP traffic won’t show its seconds-level self-correlation characteristics over session scale. Therefore, in this paper we consider splitting the raw IP flow into multiple clusters and begin the correlation analysis of the data packets in a single cluster.

For an IP flow $F = \{P_1, P_2, ..., P_n\}$, where $P_i$ represents the i-th packet in stream $F$, we use the traffic clustering algorithm to slice the stream $F$ to obtain $F = \{C_1, C_2, ..., C_k\}$, where $C_j$ represents the j-th cluster after slicing. According to the literature [5], when slicing network traffic, considering the following factors.

1. Packet arrival time. During the execution of task, the user needs to quickly send the next command after the previous command returning the result. Therefore, we define the user’s reaction time $t_r$, which is the time interval between the return of the result packet of the previous command and the packet of the latter command; At the end of a task the user need to think about the next action, and thus we define the user’s thinking time $t_t$. We assume that the process that sending request and receiving execution results of the request can be considered as a data interaction process. Therefore, it is reasonable to assume that if the time-span of a data slice is greater than the users thinking time $t_t$, then the next packet can be considered to belong to the next cluster.

2. Time interval scale. Due to the fluctuation of the network bandwidth and the like, the time-span of a packet cluster may be fluctuating, so we define the time interval scale SCALE which reduces the impact of network instability. If the arrival time interval of the adjacent data packet is greater than the SCALE times of the sum of the average intra-cluster time interval and the user’s response time $t_r$, we can assume that the latter data packet may be considered to belong to the next cluster.

3. The maximum arrival time interval. While studying some network traffic we found that if one program contains heartbeat and the heartbeat connection is frequent, especially when the interval is less than the user’s thinking time $t_t$, it’s more likely that fault slicing occur because the heartbeat packet will mislead the computer to locate the wrong last packet, so we count the maximum arrival time interval of two data packet, and the maximum arrival time interval is equal to the maximum heartbeat connection interval in this case. In a cluster, if the length of one cluster is greater than the maximum arrival time interval, and If the time interval of the adjacent data packet is greater than the attacker response time $t_r$, then the next data packet belongs to the next data packet cluster.

3.2.2. Heartbeat filter module. Considering the heartbeat package is control package and will interfere with the Trojan’s data transmission process, so we have to filter the heartbeat packet to get the pure IP data transmission flow.

The Trojan's heartbeat can complete the handshake between the controlled terminal and the controlling terminal, so that the Trojan controller can get the network environment status of the controlled terminal, which is beneficial for the attacker to continuing the attack behavior. In general, the Trojan's heartbeat is divided into two types, request response and three-way handshake. The request response type heartbeat means that the Trojan controlling terminal sends a heartbeat packet to the controlled terminal, then the controlled terminal receives the heartbeat packet and responds with a acknowledgement packet. The three-way handshake type is one step longer than the response type, to be more exact, the controlling terminal receives the acknowledgement packet of the acknowledged heartbeat of the controlled terminal, which is similar to the three-way handshake in the TCP connection.

In the Trojan design process, the heartbeat package is sent to characterize the Trojan's network survivability, and the proportion of heartbeat packets in an active Trojan connection is often higher in the entire Trojan IP stream, so in a Trojan stream there will be multiple regular Trojan heartbeat packets with same length, and we can use the association rule algorithm to extract the heartbeat packet rules and filter the heartbeat packets. In this paper, the AprioriAll algorithm [6] is used to extract the heartbeat packet rules and filter the heartbeat packets.

3.3. Traffic classification
Traffic is a time series with strong pre- and post-dependency, so we can use the bi-directional recurrent
neural network to adaptively learn the timing characteristics without manually extracting feature. Thus in this paper, we uses the bi-directional recurrent neural network as a classification model for Trojan traffic.

The Recurrent neural network is a neural network model for processing time series. On the basis of the traditional feed-forward neural network, the recurrent neural network can deal with the time series data by using neurons with self-feedback function in the hidden layer. In order to make the recurrent neural network not only calculate the state information from the previous moment, but also calculate the state information from the future moment, a bidirectional recurrent neural network is introduced, and its structure is shown in Figure 4.

![Bidirectional Recurrent neural network structure](image)

The long short-term memory network is a variant of the recurrent neural network model proposed by Hochreiter and Schmidhuber[7] for solving long-term dependence problem. On the basis of the traditional recurrent neural network, a long short-term memory network unit is composed of an input gate, a forgetting gate and an output gate is introduced to replace the hidden layer unit in traditional recurrent neural network. The structure of long short-term memory network unit is shown in Figure 5.

![Long short-term memory network Units](image)

The corresponding calculation formula is as follows.

\[
i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ei}c_{t-1} + b_i) \tag{3}
\]

\[
f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{4}
\]

\[
c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{5}
\]

\[
o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \tag{6}
\]

\[
h_t = o_t \tanh(c_t) \tag{7}
\]

Where \(\sigma\) is the sigmoid function, \(i_t\), \(f_t\), \(o_t\), and \(c_t\) represent the input gate, the forgetting gate, the output gate, and the unit state at time \(t\). In the long short-term memory network, \(i_t\), \(f_t\), \(o_t\) control the information flow. The input flow determines the input ratio. When calculating the unit state, this ratio determines the value of equation (7), and the forget gate decides whether to pass. Passing the value of the previous hidden layer output \(h_{t-1}\), the previous information transfer ratio is calculated by \(f_t\) and used to calculate the cell state \(c_t\) at time \(t\), and the output gate determines whether the hidden layer unit is passed in the output result. The value of the result \(c_t\). By using long short-term memory network units, we can better deal with the problem of gradient disappearance and gradient explosion in traditional recurrent neural networks.
4. test and result analysis
In order to verify the performance of the HTTP tunnel Trojan detection model based on bi-directional long short-term memory network model, the experimental environment used in this paper is shown in Table 1.

| Type                      | parameter                      |
|---------------------------|--------------------------------|
| Operating system          | Windows 10, 64 bit             |
| Processor                 | Intel Core i7 8700             |
| Memory                    | 16GB DDR4 2133MHz              |
| Graphics Processor Units  | Nvidia GTX 1060 6GB            |
| Tensorflow version        | Tensorflow-1.8                 |

4.1. Dataset construction
Since there is no standard dataset in the field of HTTP tunnel Trojan detection, we used the mixture of background traffic and manually collected Trojan traffic to construct its own test dataset and valid dataset to evaluate the performance of HTTP tunnel Trojan detection model. However, due to the large difference in the data size of the collection Trojan traffic and the normal Internet traffic, the normal Internet traffic volume will be much larger than the artificial Trojan traffic, which will have a greater impact on the accuracy and false positive rate of the machine learning algorithm, and it is also difficult to guarantee the purity of background traffic. To ensure reliability of system performance and the test, we used some CTU-IDS-2017 dataset[8] as the background traffic of this test, and verified the Trojan traffic detection model proposed in this paper. Two datasets are described in Table 2. The Trojan traffic of the training dataset was collected in 6 hosts in the lab LAN, 5 of them were infected with HTTP tunnel Trojans and installed Wireshark to capture the trojan traffic(including Pcshare, Gh0st and other HTTP tunnel Trojan randomly collected from the MWCollect[9] and Malfease[10], which are Trojan sample sites) and the rest one of them were used as a controlling terminal. The test dataset’s Trojan traffic was collected from 13 hosts in the lab LAN, 10 of which were infected with HTTP tunnel Trojans (including Trojan samples in some test sets and new randomly collected Trojan samples like Bifrost), and 3 were used as controlling terminal.

| Application       | Dataset            | Time              | TCP flow number |
|-------------------|--------------------|-------------------|-----------------|
| sTraining set     | Background flow    | 1day(Monday)      | 36434           |
|                   | (CTU-IDS-2017)     |                   |                 |
|                   | Trojan flow        | ×                 | 5431            |
| Test set          | Background flow    | 1day(Tuesday)     | 25964           |
|                   | (CTU-IDS-2017)     |                   |                 |
|                   | Trojan flow        | ×                 | 6352            |

4.2. Experimental evaluation metric
In this experiment, the detection rate of Trojan traffic and the false positive rate of normal traffic are used as the evaluation metric of the Trojan detection algorithm model proposed in this paper. The detection rate evaluates the detection accuracy of the Trojan detection system for Trojan traffic, and the false positive rate evaluates the ratio that Trojan detection model falsely warns user against trojan traffic. The metrics are defined as follows.

The data stream detection rate \( D \) is defined as:
The data stream false positive rate $F$ is defined as:

$$F = \frac{f_t}{n_n}$$  \hspace{1cm} (9)

Where $n_t$ represents the number of all Trojan communication data streams, $n_n$ represents the number of all normal communication traffic data streams; $t_t$ represents the number of properly identified Trojan traffic flows, and $f_t$ represents the number of normal traffic identified as Trojan traffic.

4.3. Recurrent neural network hyperparameter

Hyperparameters are model parameters established before initialization of neural network. Kalus Greff[11] proposed that during the training of neural network the influence of hyperparameters on training results and model convergence is independent after he studied the learning progress of long short-term memory network. Therefore, we adopted variable-controlling approach to fine-tune the hyperparameters like learning rate, the length of the Trojan flow slice sequence, the hidden layer size, the hidden layer number, and the batch size and find a balance point among the model size, the Trojan flow detection rate, and false positive rate. While training neural network, we used 10-fold cross validation to evaluate the capability of the Trojan detection system.

4.3.1. Learning rate. In the experiment, we kept the other variables unchanged, and continuously changed the learning rate. After 10 iterations, the results obtained are shown in Figure 6. As the learning rate decreases, the convergence rate of the Trojan traffic detection rate decreases, but the stability of the detection rate convergence increases. However, when the learning rate is less than 0.05, the stability of the detection rate convergence tends to be stable, but the convergence speed is still reduced. Therefore, the model set the learning rate to 0.05, and the Trojan detection model could be quickly converged and the convergence stability of the Trojan detection rate is maintained at a high level.
4.3.2. Trojan flow slice length. In the experiment, we continuously changed the length of the Trojan slice sequence, and the results were shown in Figure 7. As the length of the slice sequence increases, the detection rate of the Trojan flow increases first, and false positive rate gradually decreases. Then, after the slice length is greater than or equal to 6, the gradual decrease tends to be gentle. This shows that when the sequence length of the traffic session slice is equal to 6, the detection rate of the Trojan is higher, and the subsequent traffic packet length has less influence on the traffic identification capability. Therefore, the model sets the length of the slice sequence of the input traffic session to 6.

![Figure 7](image-url)

**Figure 7.** Effect of Trojan slice length on model detection performance

4.3.3. the number of hidden layers. When designing the hidden unit size of each layer, we should consider the relationship between the number of samples and the number of the parameters, because it’s widely accepted that the number of parameters should be balanced with the number of samples. So in this experiment the hidden layer sizes of this experiment were 60, 30, 24, 18 with the increasing number of hidden layer number. The relationship between the number of hidden layers and the detection rate and false positive rate is shown in Figure 8. As the number of hidden layers increases, the increasing trend of false positive rate becomes more obvious, and the detection rate of Trojan traffic gradually decreases. Therefore, we set the number of hidden layers to 2.

![Figure 8](image-url)

**Figure 8.** Effect of hidden layer number on model detection performance
4.3.4. hidden layer unit size. The relationship between the size of the hidden layer and the detection rate and false positive rate is shown in Figure 9. In the experiment, we set the hidden layer number to be 1, and we found that when the hidden layer unit is larger than 40, the detection rate of the Trojan flow tends to be gentle, but when the hidden layer unit is in the interval [40, 60], as the hidden layer unit gradually increases, the false positive rate of the model gradually decreases, and then it tends to be flat. Considering too many parameters can cause over-fitting problems in model detection, so we set the hidden layer unit size to 60.

![Figure 9. Effect of hidden layer cell size on model detection performance](image)

4.4. Results display
Through the above experiment, we established a bi-directional long short-term memory network model with two hidden layers, each of which contains 30 basic long short-term memory units. The long short-term memory network used softsign function as activation function of the fully-connected layer and used the softmax function as classification function. The loss function optimization algorithm used the Adagrad algorithm.

In the experiment, we used the above bi-directional recurrent neural network model for classification and used the dataset shown in Table 2 as data samples to train and test the Trojan detection model, and the detection performance of the Trojan detection model for some Trojans is shown in Figure 10 & Figure 11. From the figure we could see that the model proposed in this paper gained high detection rate and low false positive rate and according to Figure 11, we found that after the introduction of some new Trojans like PainRAT, the detection rate of the new Trojans did not drop significantly. The detection rate of Trojans remained above 99%, and the false positive rate of Trojan traffic remained at around 2%, indicating the Trojan traffic detection model based on sequence slices and bi-directional long short-term memory network proposed in this paper can effectively distinguish abnormal Trojan traffic and normal traffic.

![Figure 10. Data test set different categories of Trojan detection rate / false positive rate line chart](image)
As shown in Table 3, we verified the effectiveness of the proposed method by comparing the proposed model with other network-based HTTP tunnel Trojan detection techniques. From the results, we could see that compared with the original method, the Trojan detection algorithm based on the bi-directional long short-term memory network has a significant improvement on the enhancement of Trojan traffic detection rate and the reduction of normal network traffic false positive rate, which showed that our proposed model could correctly identify the malware network traffic and normal network traffic.

Table 3. Comparison of different methods of HTTP tunnel Trojan detection rate and false positive rate

| Detection method | detection rate | false positive rate |
|------------------|----------------|---------------------|
| Literature [13]  | 91.17%         | 4.56%               |
| Literature [13]  | 90.66%         | 7.28%               |
| This method      | 99.76%         | 2.33%               |

5. Conclusion
Based on the method of Trojan traffic detection utilizing sequence analysis, this paper analyzed the communication behavior of HTTP tunnel Trojan transport layer and used the associated rules algorithm to mine the Trojan heartbeat behavior and filter heartbeat to characterize the data transmission process of the HTTP tunnel Trojan. This paper used the bi-directional long short-term memory network model to obtain the HTTP tunnel Trojan traffic detection model. The results showed that compared with the existing Trojan detection model, the model could improve the detection rate of the HTTP tunnel Trojan and reduce the false positive rate of normal traffic, and during the test of the Validation dataset, we found that the method has certain ability to detect unknown Trojans. The next step we will do is to extend Trojan traffic dataset, improve the network traffic slicing performance, and find other communication behavior characteristics of the HTTP tunnel Trojan to improve the detection performance of the HTTP tunnel Trojan detection model.

Acknowledgment
We thank He Kang, Li Ding and Pan Yan for fruitful discussion. We also thank our teammates for their help with the environment building and test.

References
[1] Sun H., Liu S., Chen J. (2011) Tunnel Trojan Detection Method Based on Operation Behavior. Computer Engineering, 37:123–126.
[2] Wang Z. (2015) The Applications of Deep Learning on Traffic Identification. In: Black Hat. Las Vegas.pp.10.
[3] Wang Y., Zhou H., Feng H., Ye M. (2018) Network traffic classification method basing on CNN.
Journal of Communications, 39:14–23.

[4] Wang W., Zhu M., Zeng X., Ye X., Sheng Y. (2017) Malware traffic classification using convolutional neural network for representation learning. In: International Conference on Information Networking. Da Nang. pp.712–717.

[5] Wu S., Liu S., Lin W., Zhao X., Chen S. (2017) Detecting Remote Access Trojans through External Control at Area Network Borders. In: Symposium on Architectures for Networking & Communications Systems. Beijing. pp. 131–141.

[6] Han, J. (2005). Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. San Francisco.

[7] Hochreiter S., Schmidhuber J. (1997) Long Short-Term Memory. Neural Computation, 9:1735–1780.

[8] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A. (2018) Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: International Conference on Information Systems Security & Privacy. Maderia. pp. 108–116.

[9] Collaborative malware collection and sensing. (2014) Collaborative malware collection and sensing. http://alliance.mwcollect.org. [Accessed: 13-May-2018].

[10] Project malfeasce. (2014). Project malfeasce. https://malfeasce.oarci.net. [Accessed: 13-May-2018].

[11] Greff K., Srivastava R.K., Koutnik J., Steunebrink B.R., Schmidhuber J. (2016) LSTM: A Search Space Odyssey. IEEE Transactions on Neural Networks & Learning Systems, 28:2222–2232.

[12] Tegeler, F., Fu, X., Vigna, G., Kruegel, C. (2012) BotFinder: finding bots in network traffic without deep packet inspection. In: International Conference on emerging Networking Experiments and Technologies. Nice. pp. 349-360

[13] Li S., Yun X., Zhang Y. (2012) A Model of Trojan Communication Behavior Detection Based on Hierarchical Clustering Technique. Journal of Computer Research and Development, 49: 9–16.