INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING
DOI: 10.46300/9106.2021.15.103 Volume 15, 2021

A lightweight hybrid detection method for botnet

Wei Ma[123*], Xing Wang[4], Jiguang Wang[1], Qianyun Chen[1]
[1]School of Information Engineering, North China University of Water Resources and Electric Power,
Zhengzhou, 450046
China
[2]School of Information science and technology, Zhengzhou Normal University,
Zhengzhou, 450044
China
[3]School of Information Engineering, Zhengzhou University,
Zhengzhou, 450001
China
[4]Hangzhou Hikvision Digital Technology Co., Ltd.,
Hangzhou, 310000
China

Received: February 28, 2021. Revised: July 23, 2021. Accepted: August 10, 2021. Published: August 12, 2021.

Abstract—Botnet is a serious threat for the Internet and it has created great damage to the Internet. How to detect botnet has become an ongoing endeavor research. Series of methods have been discussed in recent research. However, one of the remaining challenges is that the high computational overhead. In this paper, a lightweight hybrid botnet detection method is proposed. Considering the features in the botnet data packets and the characteristic of employing DGA (Domain Generation Algorithm) domain names to connect to the botnet, two sensors are designed and deployed individually and parallelly. Signature detection is used on the gateway sensor to dig out known bot software and deep learning based techniques are used on the DNS (Domain Name Server) server sensor to find DGA domain names. With this method, the computational overhead would be shared by the two sensors and experiments are conducted and the results indicate that the method is effective in detecting botnet.

Keywords—Botnet, Network Security, Machine Learning, Detection.

I. Introduction

WITH the rapid development of information technology, the Internet has become indispensable in everyday life. As reported by Strategy Analytics, the number of devices connected to Internet has reached 22 billion by the end of 2018. Great economic benefits are also brought by the development of Internet. It is predicted that the digital market will rise to 4500 billion dollars in 2021. However, huge benefit carries huge risk and various network attack paradigms have yet to merge. The botnet is one kind of the most serious.

Botnet, which refers to a network consists of plenty of computing nodes infected by bot-ware and a C&C (command and control) server which is controlled by the attacker, is a huge threat to network security in nowadays [1]. With the numerous computational power, the botnet is capable of launching a massive attack on Internet. One C&C server is able to control thousands of infected computing nodes to launch massive attacks such as DDoS (Distributed Denial of Services). It is reported that in 2017, 5187 instructions were sent by C&C servers per day and 114 instructions were sent by a single C&C server. Globally, the attack from botnet has reached 28 million in a month and about 6 TB of network bandwidth was used to perform the attack activities. Furthermore, with the rapid development of the Internet of Things (IoT), IoT is becoming a new carrier of botnets. Mirai, which is the most famous IoT botnet, used to make the most severe DDoS attack in 2016 [2]. Billions are lost with the attacks launched by botnets.

Therefore, it is important to identify botnet victims in the Internet. To address the threats brought by botnets, researchers have done a lot of research. Network traffic or network flow characteristics are most popular for botnet detection. Due to the bots in botnet exploit network to communicate and negotiate, some malicious activities are hiding in network traffic. The key to detect botnet is to find out the features of malicious activities. For example, N-EDPS system exploits traffic signatures to detect known bot software by checking the outbound traffic [3]. This system can be integrated into Snort, an open-sourced intrusion detection system. Another way of using network traffic is anomaly detection. Soniya and Wilscy developed a method which can filter normal traffic from all network traffic and dynamic pattern analysis will be applied to the rest traffic to discover suspicious network activities [4]. BotMiner is a 2-step botnet mining and detecting system [5]. In the first step, BotMiner would filter some traffic based on some learned knowledge out of all network traffic and clustering will be applied to the rest traffic. In the second step, anomaly detection is carried out on the clusters generated in step one. Similar features between clusters are used to discover botnet. Fedynyshyn et al. proposes a method which classifies types of C&C messages [6]. Features are extracted from different types of messages and they are used to detect botnet. This method doesn't need to check the payload of the data packet, hence it can detect encrypted C&C traffic. Considering the characteristic that the wide application of HTTP in C&C servers, Xiong et al. designed a botnet detection system based on HTTP [7]. This system will verify every web request and only authorized websites are allowed to visit. This system will prevent terminal nodes from connecting to the C&C server, but it's only useful for HTTP based botnet. Wurzinger et al. proposes a prior knowledge free method to detect botnet [8]. By calculating the similarity between normal network traffic and abnormal traffic, this method is able to find out the deviation of abnormal traffic. Alike abnormal traffic will be located and analyzed to check it is botnet traffic or not. And recently, ML (Machine Learning) based methods were adopted in botnet detection with network traffic or network flow features [9]. For example, graph-based methods for botnet detection and feature clustering [10], [11]. Deep packet inspection (DPI) was integrated with ML to detect botnet [12]. And deep learning was adopted for a hybrid detection method in IoT environment [13].

Most of the network traffic based botnet detecting method need prior knowledge to detect known bot software. Only a few methods are prior knowledge free but the performance of false positive rate and false negative rate are poor.

Another kind of botnet detecting method is DGA (Domain Generation Algorithm, DGA) based. Bot software always uses dynamic domain names to tell victim hosts how to connect to C&C servers and DGA is used to generate those domain names. In the early time, researchers would submit the resolved DGA domain name to some ranking site such as malware domain list. However, DGA is able to generate thousands of domain names but only a few of them are picked out for bot software to make use of. Methods are developed to detect DGA domain names. Antonakakis et al. uses the combination of clustering algorithms and classification algorithms to detect known DGA domain names and dig out unknown domain names [14]. Kazumichi et al. exploits DNS (Domain Name Server) traffic and blacklist to detect unknown malicious domain names [15]. It assumes that when an unknown domain appears frequently with a known malicious domain name at the same time, then this unknown domain name should be malicious as well. Sharifnya and Abadi proposed a reputation based botnet discovery method by calculating an activity matrix for every DNS request, and the hosts with the lowest reputation will be identified as botnet hosts [16]. Lee and Lee uses failed DNS request data to discover unknown types of the botnet [17]. DGA based methods and meanwhile, to monitor DNS activities is a heavyweight task for the host to detect botnet.

However, there are still some flaws for both kinds of the methods. Firstly, for traffic or flow based methods, prior knowledge is needed to detect known bot software. Only a few methods are prior knowledge free but the performance of false positive rate and false negative rate are poor. Secondly, for DGA based methods, they always based on DNS activities and multiple dimensions of data are used but domain names themselves are not getting enough attention. And thirdly and most importantly, the computational overhead of current methods is high due to that monitoring no matter network traffic or DNS activities is a heavyweight task for the host. Considering the imperfection of current methods, we propose a lightweight hybrid detection method for botnet by combining the traffic base method and DGA based method in this paper. This method takes advantage of network traffic and DGA domain name and the contributions of this paper are:

A method is designed with which we examine the outbound network traffic to discover known bot software based on prior knowledge.

DNS requests will be resolved to find out abnormal domain names, and the two steps are deployed on different positions to reduce the computational overhead and consequently make the method lightweight.

Experiments are conducted and the performance is evaluated for the proposed method.

This paper is organized as follows: The first section introduces the background and promotions with the working patterns of botnet in section 2. The design principles and overall design are discussed in section 3. In section 4, we introduce detailed implementations and evaluated the method with experiments. In the end, the conclusion is given in section 5.

II. Working patterns of the botnet

Despite the diversity of botnet or bot software, there are some mutual working patterns. A botnet controller, which is an attacker or hacker, always controls the botnet in a unique way. The typical architecture of a botnet is shown in Figure 1, and there are four stages in the control flow.

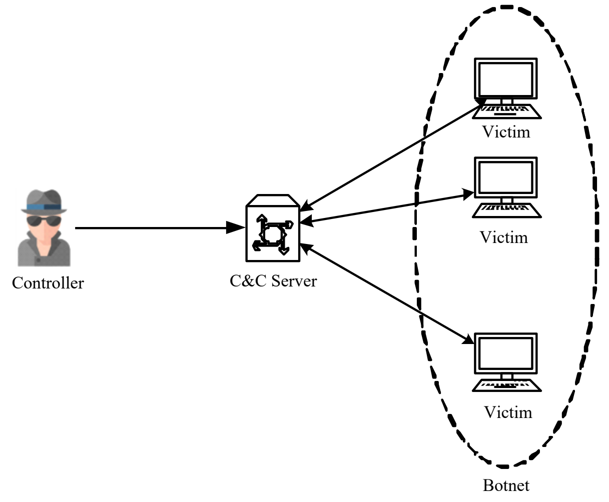

Fig. 1. Botnet architecture

In stage 1, the victim host will be infected via some vulnerability and binary applications of botnet will be download by some script written by the controller.

In stage 2, once the bot software is installed in the victim host, the victim will try to connect to the C&C server initiatively.

In stage 3, when the victim connects to the C&C server successfully, the registration process will be done, which means that the victim has joined the botnet officially.

In stage 4, the controller is able to send commands to the registered victim host via the C&C server, such as launching an attack and updating bot software.

The C&C server plays an important role in botnet architecture. The controller doesn't control or send commands to the botnet directly but employ C&C server as a broker. Once the connections between the botnet and C&C server failed, the controller would lose control to the botnet. There are several kinds of C&C connection structures, central structure, distribute structure and hybrid structure, as shown in Figure 2.

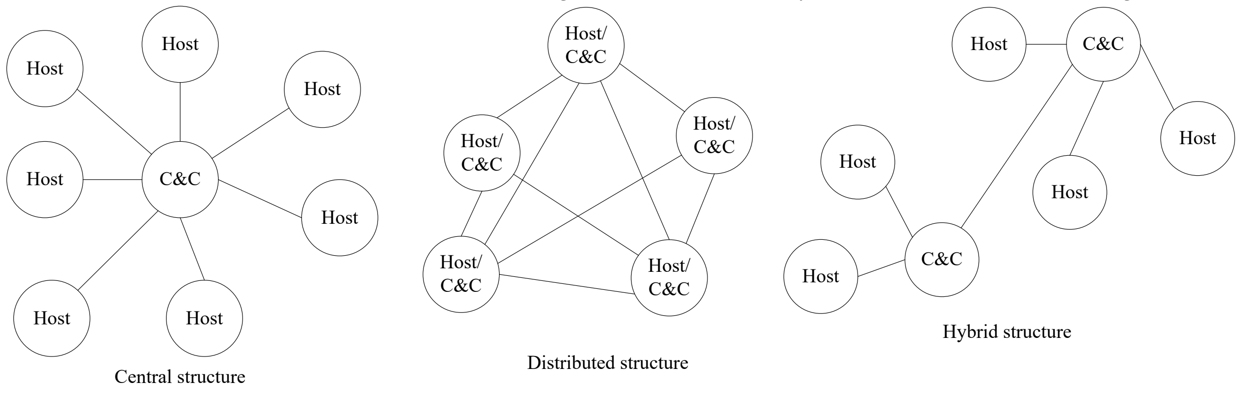

Fig.2. C&C structures

The central structure is simple and no special communication protocols are required. Every victim host connects to the C&C server directly with some universal protocol, such as IRC (Inter Relay Chat) and HTTP. The character of easy-to-use makes central structure the most popular for a lot of bot software such as Zeus and conficker. The biggest disadvantage of the central structure is lacking robustness. Due to all hosts rely on one single C&C server to communicate with the controller, the entire botnet would be destroyed when the C&C server is dug out. Relatively the distributed structure is more complex while every host in the botnet is one C&C server at the same time. In other words, it can be considered that there is no C&C server in distributed structure or everyone is a C&C server. In distributed structure, each host should connect to at least one another host and the message will be broadcasted to entire network through the support of P2P (Peer-to-Peer) protocol. Robustness can be fulfilled easily in distributed structure but there are also limitations. The complexity of the network will significantly increase when more hosts join the network which brings many more connections, which leads to high latency and unreliable commands transmission. The hybrid structure is the combination of the central structure and distributed structure, where multiple C&C servers are connected with each other and one C&C server takes charge of communicating with one section of hosts. This structure combines the advantages of the other two structures.

No matter what structure the botnet employs, it is important for hosts to find and connect to the C&C server. Addresses of C&C servers are hard-coded in bot software in early age while DGA is always adopted in nowadays, due to hard-coded addresses are easy to be dug out in network traffic and not flexible enough. DGA will generate random domain names with different input which is mostly the system time. To add confusion, only a few of the generated domain names will be registered in the DNS server and the rest of them will be abandoned. Brought by the confusion, it would be hard for the security to find out malicious domain names used to connect to the C&C server. Even malicious domain names are successfully located, it will be easy to generate a new batch of domain names to get rid of restrictions. Hence, it is critical to detect DGA domain names when hosts make a DNS request.

Another working pattern of botnet is network traffic itself. As mentioned above, only a small number of bot software uses private protocols, and most bot software uses common protocols such as IRC and HTTP [18, 19]. The benefits of using common protocols are easy to hide because private protocols are easier to be discovered due to the rareness of new protocols. However, some signatures of botnet may be hidden in protocol communications. To avoid signature extraction, bot software always utilizes techniques encrypting, hiding or confusing the payload of communications [20, 21]. Therefore, it's necessary to confront those techniques when analyzing network traffic.

III. Design of the Proposed Method

A. Design principles

The botnet detection method proposed by this paper should be able to detect botnet traffic and be designed based on the following principles.

Real-Time: The method should detect botnet or bot activities with very low latency. The principle of real-time indicates that the method should be able to complete detection in limited time.

Reliability: The method should detect botnet from different dimensions. Sensors, or checkpoints, should be deployed in different places to avoid single point failure.

Economy: The method should not add too much overhead to the hosts due to the computational power of the hosts may be limited.

B. Architecture and overall design

The method aims to detect the bot activities when hosts are trying to connect to C&C servers by employing both DGA detection and network traffic detection. To fulfill the principle of reliability, DGA detection and network traffic detection should be deployed in different places in the network. Focusing on the domain names, DGA detection could be deployed on the domain name server while network traffic detection could be deployed on the network gateway or router because it inspects the data packet. The architecture of the proposed method is illustrated in Figure 3.

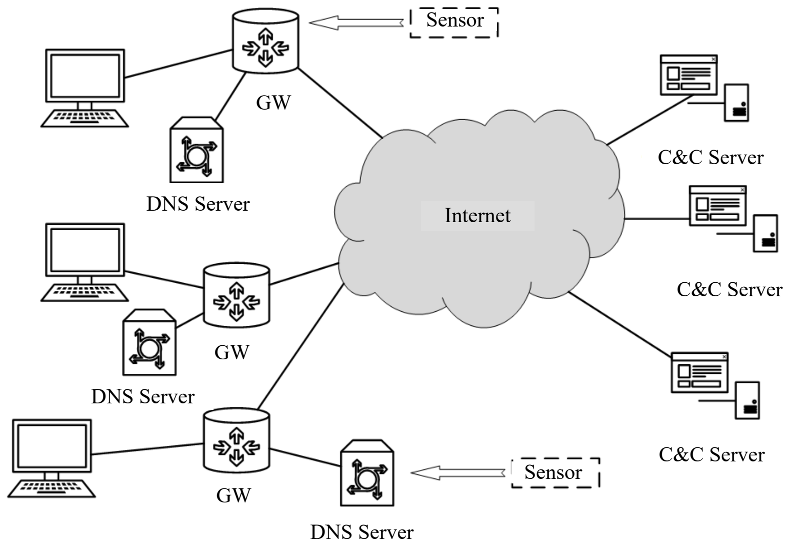

Fig. 3. The architecture of the proposed method

With this logical architecture, the host connects to the Internet via a gateway (GW). And the host also requests DNS service using the same gateway. What is noteworthy is that GW and DNS servers in this architecture are logical components, which means that they can be independent components and integrated or even integrated with the host. However, for the reason of economy, we design them with two independent components in the proposed method. Two different sensors are deployed in GW and DNS servers, sensor for network traffic and sensor for DNS server.

Sensor for network traffic (GW Sensor)

The sensor for network aims to detect and analyze the network traffic. As the only data channel for the host, GW is able to capture every data packet sent from the host. Hence the GW sensor works like a packet filtering firewall and examines every data packet when the packet arrives at the GW to fulfill the principle of real-time. The network traffic sensor detects not only layer-3 and layer-4 (network layer and transport layer in TCP/IP architecture) but also layer-5 (Application layer) traffic. Note that most bot software uses common protocols, we exploit signature detection to find out known bot activities. Furthermore, to avoid the impact from hiding or confusing techniques and relieve the computational pressure, payload-free signatures are adopted in the method. The signatures are extracted from two specific network packets, the initial packet of the network flow and the attack command packet.

Initial packet, which is the first packet sent from victim host to the C&C server. It is used to register the victim on the C&C server.

Attack command packet, which is used as the command sent from the C&C server. Following a simple protocol, the victim will launch an attack according to the attack command packet.

The features extracted from the two kinds of packets are able to identify the know bot software with the prior knowledge. The specific features will be discussed in the implementation section.

Sensor for DGA (DNS Sensor)

Based on the character that the host always uses DGA to find and connect to the C&C server, the DNS sensor aims to intercept every DNS request and check the target domain name malicious or not. The generated domain names, which also consist of letters and digits as normal domain names, always present some character characteristics, such as length, the number of special characters, the ratio of digits to letters, etc. These characteristics can be used for feature extraction in machine learning algorithm. In our method, a deep learning based scheme is used in the proposed by exploiting the long-short term memory (LSTM) neural network. LSTM is a special type of recurrent neural network which is able to capture time-sequenced features of the input. In LSTM, 3 gates are introduced to control the states of neurons. The first gate, forget gate, controls how much state C_{t-1} in the last time will be reserved to the state C_t in current time:

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right]\right) + b_f)$$

where x_t is the current input and h_{t-1} is the output from the last hidden layer. W_f and b_f are trainable parameters in the neural network. The second gate, input gate, determines what information from the input should be updated into the neurons' state:

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right)$$

$$\tilde{c}_t = \tanh\left(W_C \cdot \left[h_{t-1}, x_t\right] + b_C\right)$$

And the update process is:

$$C_t = f_t * C_{t-1} + i_t * \tilde{c}_t$$

The last gate, output gate, is responsible for generating the output value h_t based on h_{t-1} and x_t:

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

With those three gates LSTM is able to make up for some flaws of traditional recurrent neural network such as remembering long term information of the input. Meanwhile, comparing with machine learning algorithms, LSTM is a deep learning algorithm which is able to extract features automatically without feature engineering. Even the features are changed, LSTM is still capable of correcting and adopting. With sufficient training set (domain simples), it would be more accurate and more efficient for this deep learning method than traditional machine learning methods.

LSTM is adopted to train a classifier to distinguish the normal domain names and the DGA domain names. In the proposed method, the classifier is pre-trained offline with collected DGA domain names and the detection process is online to reduce the computational overhead in real-time. The performance of the classifier is important, and the performance is evaluated from a set of metrics in the implementation and evaluation section.

IV. Implementation and evaluation

A. Workflow of the method

With the proposed method, to ensure the principle of reliability, the two sensors would work individually and parallelly. No matter the victim host is sending bot network packets or requesting to visit DGA domain names, sensors will be able to detect and respond accordingly. The workflow of the proposed method is shown in Figure 4.

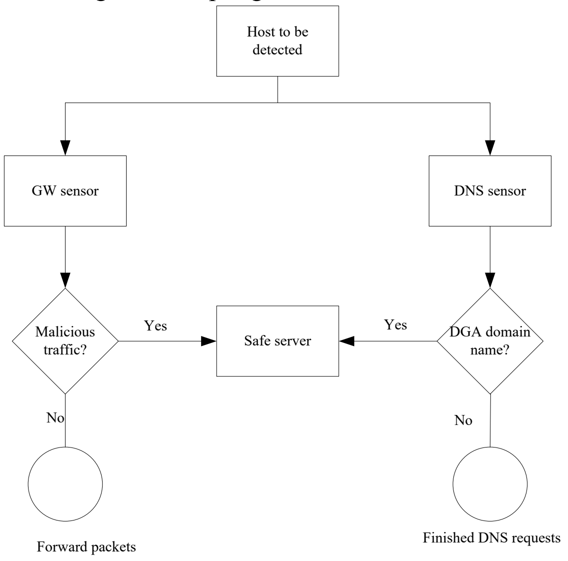

Fig. 4. Workflow of the method

There are 6 steps in the work flow:

The host to be detected would send outbound network traffic, may be data packets, DNS requests, or both.

If the outbound network traffic contains data packets, the packets will be sent to the GW where the GW sensor can capture and analyze them.

Meanwhile, if the outbound network traffic contains DNS requests, the DNS sensor deployed on the DNS server will be able to capture the requests and analyze them.

The features of known bot traffic are hardcoded in the GW sensor. When the features match the captured network traffic, a "malicious traffic" trigger will be pulled.

The trained LSTM classifier is deployed within the DNS sensor and every DNS request would be determined as requesting a DGA domain name or not. When a DGA request is discovered, a "DGA request" trigger will be pulled and the request will be reformed.

A "safe server" is deployed in the network. The safe server is a web server with a simple warning webpage. The trigger information, such as "malicious traffic" and "DGA request" would be sent to this safe server by the sensors and the warning information would be given by the safe server.

Moreover, because the two sensors work in parallel, the detection performance of the method should be implemented and evaluated separately.

B. Implementation and evaluation of GW sensor

We analyzed three bot software families which are popular in China, "xingtian", "nightguard" and "network nuclear weapon", to verify that whether the signatures of the data packet can be used to detect botnet or not. In our experiments, we captured the initial packets when the victim host was connecting to C&C server for the first time and command packets sent from C&C server. Those packets were analyzed to find patterns to detect corresponding bot software. Due to that different bot software launches different types of attacks, experiments and analyses were made individually.

Xingtian

Firstly, we captured and analyzed the features of initial packets of xingtian, as shown in Table 1.

Table 1. Features of xingtian's initial packets

Indicator	length	Offset in the payload
Host OS	64	4
Memory size	32	68
CPU frequency	32	100
Bandwidth	32	132

As we captured, the total length of the initial packet is 238 bytes in which 184 bytes are payload carrying the information including host operating system, memory size of the host, CPU frequency of the host and the network bandwidth of the host. Furthermore, a significant signature of the initial packet is at the beginning of the initial packet with the content "b0 00 00 00 77 00 00 00 04 08 00 00".

Secondly, xingtian is able to launch attacks from two layers, transportation layer such as TCP-SYN attack and UDP flooding, and application layer such as infinite CC. Hence, we captured and analyzed the command packets, summarized in Table 2.

Table 2. Features of xingtian's command packets

Type	Minimize packet length	Minimize payload length	Offset in payload
TCP-SYN	86	32	4
UDP flooding	86	32	4
Infinite CC	92	38	4

Different from the initial packets, the command packets are with variable length because the attack payload, which is not with a fixed length, is integrated with the packet. For TCP-SYN attack and UDP flooding attack, the minimum payload is 32 bytes and the minimize is 86 length, while the infinite CC is with the numbers of 38 and 92. The attack identifier, which is also integrated with the payload, starts from the offset value of 4 and with a 4 bytes length.

Nightguard

The length of the initial packet of nightguard is 834 bytes and in which 780 bytes is payload. Unlike xingtian, nightguard only collects the information of host OS with an offset value 4 in the payload. What is similar is that nightguard also has a signature at the beginning of the packet with the content "55 55 09 00". Night performs attacks of TCP-SYN, UDP flooding and infinite CC as well, and the features of nightguard are summarized in Table 3.

Table 3. Features of nightguard

Type	Packet length	Payload length	Offset in payload
Initial	834	780	0
TCP-SYN	834	780	416
UDP flooding	834	780	416
Infinite CC	834	780	640

The length of command packets of nightguard is fixed as 834 and so is the payload with a length of 780. Hence the packet length can be used as a feature to detect nightguard.

Network nuclear weapon

The length of the initial packet of network nuclear weapon is 1090 bytes with a 1036 bytes payload including host OS, memory size, CPU frequency and host date and time. However, the beginning of the initial packet is not constant. Network nuclear weapon is also able to launch attacks of TCP-SYN, UDP flooding and infinite CC, and we collected the features in Table 4.

Table 4. Features of network nuclear weapon

Type	Packet length	Payload length	Offset in payload
Initial	1090	1036	0
TCP-SYN	1090	1036	136
UDP flooding	1090	1036	136
Infinite CC	1090	1036	136

The packet length and payload length are stable with network nuclear weapon, which means that packet length can be used as a feature as well.

Based on the features and signatures we collected, experiments were taken to evaluate our method. The features we used in the experiments include:

Signature in the initial packet;

Length of the initial packet;

Length of the commands packet;

Content of the targeted offset in the payload.

Due to that, not all features are applied to all bot software, we only made use of parts of them when some feature was missing in the bot software to be detected. The detection was implemented with flask and WebSocket. Results of the experiments are shown in Figure 5.

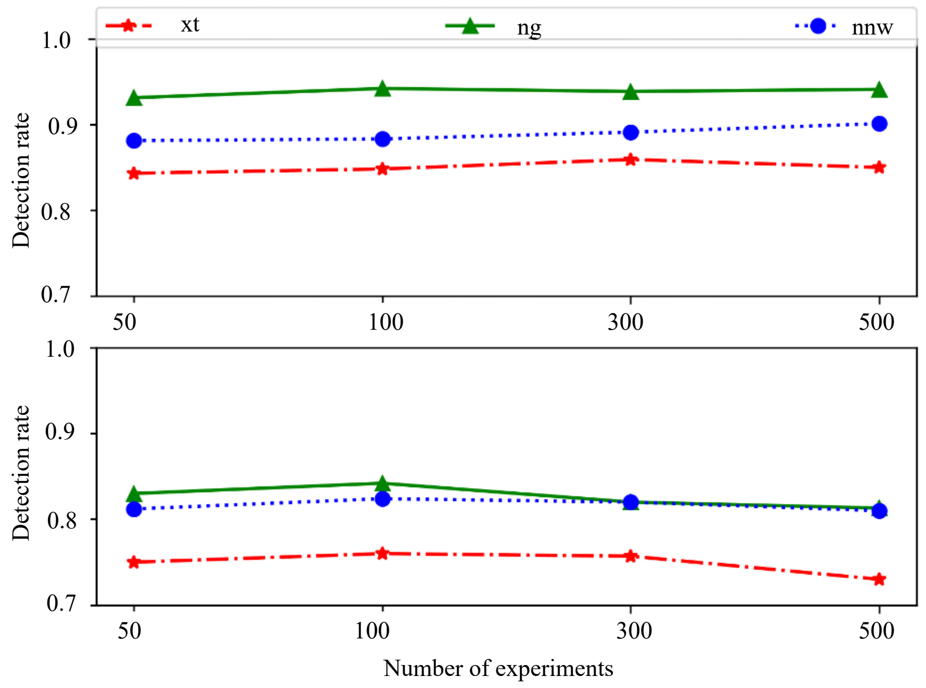

Fig. 5. Result of experiments on GW sensor

We designed two sets of experiments. In both sets, every bot software was installed and commanded to launch different types of attacks for 50, 100, 300 and 500 times to calculate the detection rate in different situations. The difference between two sets was that in the first set only the bot software was allowed to connect to the Internet in the infected host while there is some other software were connecting to the Internet to make confusion in the second set. Due to that strict comparison were conducted on the data packets based on the features, the detection rate was the only evaluation metric. The results show that firstly, different software benefited from different numbers of features. Xingting (xt) only took advantage of 2 features, so the detection rate was the lowest in all three. Nightguard (ng) took advantage of all 4 features and the detection rate was the highest while the network nuclear weapon (nnw) was in the middle. Secondly, there was a slight change when the attack times were different as shown in the figure. Thirdly, the detection rate would change when the running environment changed. Comparing with the first set, because of the existence of confusion programs, the detection rate of the second set decayed significantly. It is clear that using GW sensor alone is not enough for detecting botnet, and DNS sensor is employed to make up for the flaws.

C. *Implementation and evaluation of DNS sensor*

We adopted an LSTM neural network to realize the DNS sensor. LSTM is able to take strings with variable length as an input without feature engineering. It will capture the sequential characteristics hidden in the domain name strings, and with which we would be able to distinguish DGA domain names from normal domain names.

The structure of LSTM adopted in the experiments was simple, as illustrated in Figure 6. Each character in the input domain name would be encoded into a vector, and in the experiments, this part of work was done by an embedding layer which is the first layer of the LSTM neural network. In the embedding layer, the variable-length input will be encoded into a vector with a fixed length of 128. The second layer of the structure was the LSTM layer, the core layer in the structure. LSTM layer took the 128-dimension vector generated from the embedding layer as input, and generate a 1-diemension vector for the next layer. The final layer, dense layer, took the output from the LSTM layer and generated a value between 0 and 1 which was calculated by a sigmoid function. The output value of the dense layer indicated the probability that the input domain name being a DGA domain name.

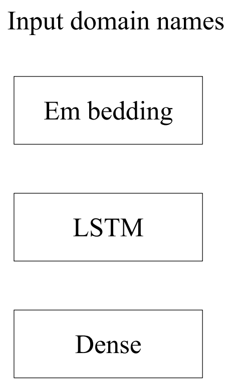

Fig. 6. LSTM structure in the experiments

The experiments were conducted based on a self-collected dataset, includes 1 million normal domain names gathered from Alexa and 100 000 DGA domain names gathered from the Internet. 5% domain names of the dataset were held for cross-validation and the rest of them were used to train the neural network. It's a binary classification problem to determine a domain name DGA domain name or not, hence the following metrics were used to evaluate the neural network.

(1) Precision

Precision is used to describe the proportion of true positive samples to all predicted positive sample, which consists of false positive samples and true positive samples. The equation for calculating precision is:

$$Precision = \frac{\sum TruePositive}{\sum TruePositive + \sum FalsePositive}$$

(2) Recall rate

The recall rate is used to describe the proportion of true positive samples to all positive samples. It is calculated with:

$$Recall = \frac{\sum TruePositive}{\sum TruePositive + \sum FalseNegative}$$

(3) F1-score

F1-score is the harmonic mean of precision and recall rate to balance them. It is a useful metric to evaluate the model, calculated with:

$$F_1 = 2 \bullet \frac{Precision \bullet Recall}{Precision + Recall}$$

(4) Receiver operating characteristic curve, ROC curve

ROC (Receiver operating characteristic) curve utilizes both true positive rate and false positive rate to evaluate the classifier. True positive rate (TPR) is the proportion of true positive samples to all positive samples, calculated with:

$$TPR = \frac{\sum TruePositive}{\sum TruePositive + \sum FalseNegative}$$

False positive rate (FPR) is the proportion of false positive samples to all negative samples, calculated with:

$$FPR = \frac{\sum FalsePositive}{\sum FalsePositive + \sum TrueNegative}$$

Using FPR as horizontal axis and TPR as vertical axis, dynamically adjusting TPR and FPR, a curve will be formulated which is the ROC curve. To evaluate the ROC curve, AUC (Area Under the Curve) is introduced. AUC is a value between 0 and 1. The bigger the AUC is, the better the classifier will be.

We implemented the experiments with Keras. After 100 epochs of training, the result was collected in Table 5.

Table 5. Evaluation for LSTM neural network

Type	Precision	Recall	F1-score	Amount
Normal	0.9771	0.9787	0.9779	21986
DGA	0.9787	0.9771	0.9779	21988
Avg/tot	0.9779	0.9779	0.9779	43974

al

In the cross-validation, the volume of the test dataset is 43 974, in which 21 986 samples are normal domain names and 21 988 are DGA domain names. The results of precision, recall rate and F1-score indicated that the performance of the neural network was good. Furthermore, we also collect the numbers of false positive samples and false negative samples, and with those data, the ROC curve was illustrated in Figure 7.

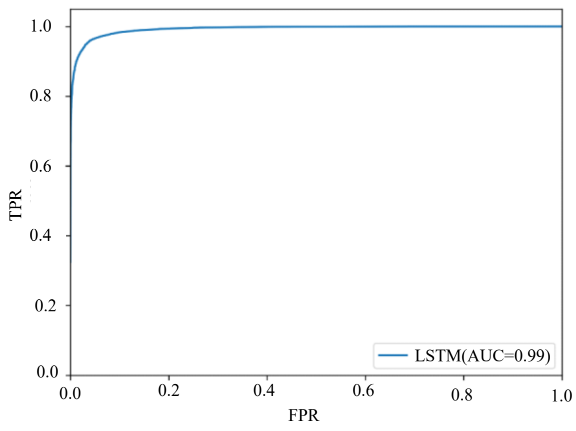

Fig. 7. ROC curve of the experiments

In the experiments, the numbers of true positive, false negative, false positive and true negative are 21519, 467, 504 and 21 484. And accordingly, the AUC of the ROC curve is 0.9931. With the combination of DNS sensor and GW sensor, it would be more efficient and reliable for the proposed method.

D. Discussion

The discussion is based on the the design principles proposed in section 3.1.

Discussion on real-time principle: Although the training process would take some time, it is offline. The detecting process is almost real-time. For the GW sensor, it would extract simple features such as packet length and payload length from the outbound network traffic, and the time consumption is little. For the DNS sensor, with the trained LSTM classifier, the classification process can determine the requested domain name malicious or not immediately. Therefore, the real-time principle is fulfilled.

Discussion on reliability principle: The GW sensor and the DNS sensor are deployed on two different locations which are separated from each other logically. Hence, the two sensors would work individually and without overlap between them. It is clear that even one of the sensors fails or crashes, the other one would be functioning without influence. The single point failure can be avoided with the proposed method and the principle of reliability can be assured.

Discussion on economy principle: With the proposed method, tasks such as data collection, analysis and detection are performed by the sensors, and the hosts in the network won't take any additional computational overhead. And in fact, the entire detecting process is transparent for the hosts. Hence, any kind of hosts, no matter generic computers or IoT devices which are resource constrained, is applicable to the prosed method. And meanwhile, the logical separation of two sensors also reduces the computational overhead of the individual sensor, and that is the reason that we call this method “lightweight”. The economy principle is also fulfilled.

Furthermore, the availability of the proposed method should be discussed as well. The performance of GW sensor is shown in Figure 5 and the performance of the DNS sensor is shown in Figure 7. It can be seen that the performance of GW sensor is not so good. However, note that the proposed method is a two-armed method, which means the overall performance depends not only on one sensor but depends on both sensors. The performance of DNS sensor is good, and the hybrid method combing the two sensors would achieve a good performance as well.

V. Conclusion

As a huge threat on the Internet, botnets have gained a lot of attention. Based on the principles of real-time, reliability and economy, a lightweight hybrid detection method is proposed in this paper. Two sensors are deployed separately on network gateway and DNS server. Features extracted from data packets are employed for gateway sensor and deep learning technique are adopted for DNS sensor. Results of the experiments indicate that the method is efficient.

Acknowledgements

This paper is supported by Henan Programs for Science and Technology Development (182102210329); Henan Key Research Projects of Universities (20A413008) and Legal Theory Research Project in Justice Department of the People’s Republic of China (17SFB3020).

References

[1] S.S.C. Silva, R.M.P. Silva, R.C.G. Pintob, et al., "Botnets: A survey," Computer Networks, vol. 57, no. 2, pp.378-403, Feb. 2013.
[2] M. Antonakakis, T. April, M. Bailey, et al., "Understanding the Mirai botnet," SEC'17: Proceedings of the 26th USENIX Conference on Security Symposium, pp. 1093-1110, 2017.
[3] S. Behal, S.B. Amanpreet, K. Krishan, "Signature-based botnet detection and prevention," Proceedings of International Symposium on Computer Engineering and Technology, Panjab University, Chndiagarh, 2010.
[4] B. Soniya, M. Wilscy, "User traffic profile for traffic reduction and effective bot C&C detection," IJ Network Security, vol.16, no. 1, pp. 46-52, 2014.
[5] G. Gu, R. Perdisci, J. Zhang, et al., "BotMiner: Clustering analysis of network traffic for protocoland structure-independent botnet detection," 17th USENIX Security Symposium, pp. 139-154, 2008.
[6] G. Fedynyshyn, M.C. Chuah, G. Tan, "Detection and classification of different botnet C&C channels," International Conference on Autonomic and Trusted Computing. Springer, Berlin, Heidelberg, 2011.
[7] H.J. Xiong, P. Malhotra, D. Stefan, et al., "User-assisted host-based detection of outbound malware traffic. International Conference on Information and Communications Security," Springer, Berlin, Heidelberg, 2009.
[8] P. Wurzinger, L. Bilge, T. Holz, et al., "Automatically generating models for botnet detection," European symposium on research in computer security. Springer, Berlin, Heidelberg, 2009.
[9] A. Nazir, R.A. Khan, "Network intrusion detection: Taxonomy and machine learning applications," Machine Intelligence and Big Data Analytics for Cybersecurity Applications. Springer, Cham, pp. 3-28, 2021.
[10] A.A. Daya, M.A. Salahuddin, N. Limam, et al., 2020. BotChase: Graph-based bot detection using machine learning. IEEE Transactions on Network and Service Management, vol. 17, no.1, pp. 15-29, 2020.
[11] S. Chowdhury, M. Khanzadeh, R. Akula, et al., "Botnet detection using graph-based feature clustering," Journal of Big Data, vol. 4, no. 1, pp. 1-23, 2017.
[12] W. Wang, Y.Y. Shang, Y.Z. He, et al., "BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors," Information Sciences, vol.511, pp.284-296, 2020.
[13] S.I. Popoola, B. Adebisi, M. Hammoudeh, et al., "Hybrid deep learning for botnet attack detection in the internet-of-things networks," IEEE Internet of Things Journal, vol. 8, no. 6, pp.4944-4956, 2020.
[14] M. Antonakakis, R. Perdisci, Y. Nadji, et al., "From throw-away traffic to bots: Detecting the rise of DGA-based malware," Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12), 2012.
[15] S. Kazumichi, I. Keisuke, T. Tsuyoshi, et al., 2012. "Extending black domain name list by using co-occurrence relation between DNS queries," IEICE Transactions on Communications, vol. 95, no. 3, pp. 794-802, 2012.
[16] R. Sharifnya and M. Abadi, "A novel reputation system to detect DGA-based botnets," International Econference on Computer & Knowledge Engineering IEEE, pp. 417-423., 2013.
[17] J. Lee, H. Lee, "GMAD: Graph-based malware activity detection by DNS traffic analysis," Computer Communications, vol. 49, pp. 33-47, 2014.
[18] J. Nazario, "Blackenergy Ddos Bot Analysis," Arbor,2007.
[19] D. Plohmann, E. Gerhards-Padilla, "Case study of the miner botnet," 2012 4th International Conference on Cyber Conflict (CYCON 2012). IEEE, 2012.
[20] S. Specht and R. Lee, "Taxonomies of distributed denial of service networks, attacks, tools and countermeasures," CEL2003-03, Princeton University, Princeton, NJ, USA, 2003.
[21] N. Falliere, "SALITY: Story of a peer-to-peer viral network," Rapport technique, Symantec Corporation 32, 2011.

Wei Ma, born in December 1985, male, doctor, lecturer. He graduated from Henan Normal University with a bachelor's degree in computer science and technology in 2008; He graduated from Beijing Jiaotong University with a master's degree in information security in 2011; He graduated from Beijing Jiaotong University in 2016 with a doctorate in information security. He is currently a postdoctoral researcher with Zhengzhou University and Zhengzhou Normal University, and a teacher with North China University of Water Resources and Electric Power. He has published 20 articles in international journals and conferences. His research interests include trusted computing, IoT security and cloud computing.

Contribution of individual authors to the creation of a scientific article (ghostwriting policy)

Botnet poses a serious threat to the Internet and causes great harm to the Internet. How to detect botnet proposed by Wei Ma and Xing Wang has become an ongoing research. They proposed a lightweight hybrid botnet detection method. Jiguang Wang and Qianyun Chen designed two sensors according to the characteristics of Botnet packets and the characteristics of using DGA (domain generation algorithm) domain name to connect botnets. They did experiments and concluded that this method can effectively detect botnets. Wei Ma wrote the first draft, and Xing Wang and Jiguang Wang reviewed and edited the article. All authors have read and agreed to the publication of the manuscript.