EasyChair Preprint

№ 8084

GBLNet: Detecting Intrusion Traffic with Multi-Granularity BiLSTM

Li Wenhao and Xiao-Yu Zhang

May 24, 2022

GBLNet: Detecting Intrusion Traffic with Multi-granularity BiLSTM

Wenhao Li[1,2] and Xiao-Yu Zhang[1,2(✉)]

[1] Institute of Information Engineering,
Chinese Academy of Sciences, Beijing 100093, China
{liwenhao,zhangxiaoyu}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100093, China

Abstract. Detecting and intercepting malicious requests are some of the most widely used ways against attacks in the network security, especially in the severe COVID-19 environment. Most existing detecting approaches, including matching blacklist characters and machine learning algorithms have all shown to be vulnerable to sophisticated attacks. To address the above issues, a more general and rigorous detection method is required. In this paper, we formulate the problem of detecting malicious requests as a temporal sequence classification problem, and propose a novel deep learning model namely GBLNet, girdling bidirectional LSTM with multi-granularity CNNs. By connecting the shadow and deep feature maps of the convolutional layers, the malicious feature extracting ability is improved on more detailed functionality. Experimental results on HTTP dataset CSIC 2010 demonstrate that GBLNet can efficiently detect intrusion traffic with superior accuracy and evaluating speed, compared with the state-of-the-arts.

Keywords: Intrusion Detection · Network Security · Model Optimization.

1 Introduction

It has been a seesaw battle between intrusion traffic utilization and detection for decades. With the rapid development of network technology, many commercial applications are now transiting to a lightweight browser/server model (B/S). With such model, information is transported from a directory service via Hyper Text Transport Protocol (HTTP) or HTTP via HTTP over TLS (HTTPS), where TLS is an encryption protocol on transport layer. Although encryption technology has been popularized in recent years, it is argued that HTTP, with plaintext transmission, still dominates the intrusion traffic [9]. Most attackers who launch attacks on web applications pass the HTTP request method. As announced in [2], 80% of the Open Web Application Security Project (OWASP) top 10 network attacks are based on the HTTP, which lead to the vulnerability of servers and the leakage of user privacy data. It is more efficient to deploy a

HTTP based intrusion detection system than to repair a large number of web application vulnerabilities.

Motivated by the sensitive advantages of Bidirectional Long Short-Term Memory (BiLSTM) in temporal text processing and Convolutional Neural Networks in feature extracting, we take the temporal sequence classification problem into account to propose a novel deep learning model by girdling BiLSTM with multi-granularity convolutional features to detect malicious requests. It is worth mentioning that the model has greatly improved the convergence speed and convergence speed, which promotes the deployment of real-time updating dynamic intrusion detection systems. The main contributions of our work are as follows:

- A novel intrusion detector, namly GBLNet, is proposed to detect malicious HTTP request. GBLNet considers multi-granularity features extracted by Convolutional Nets to enhance the power of bidirectional LSTM.
- Encouraged by the enhanced CNNs-girdled BiLSTM, GBLNet shows superior efficiency when detecting malicious network traffic, compared with the existing detectors.

2 Detection with Convolution-Girdled BiLSTM

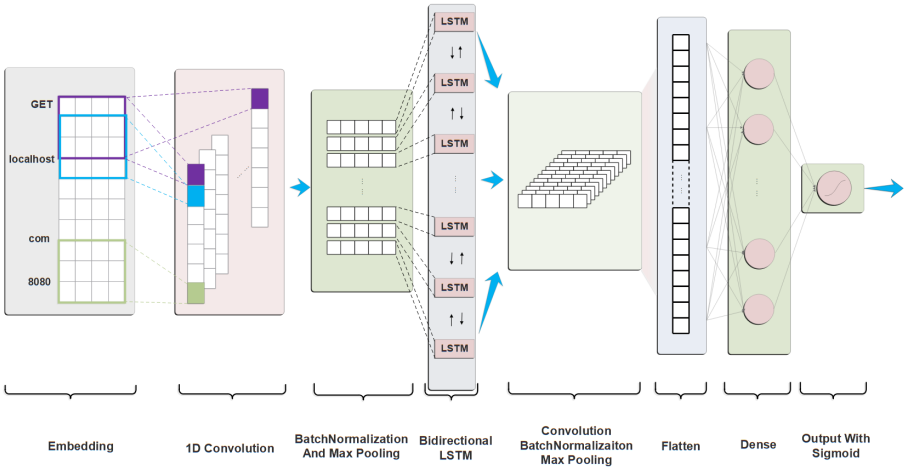

Fig. 1. Structure of GBLNet.

2.1 Detail of Model

We apply the Embedding layer as the first layer of our model. The Embedding layer can be divided into two parts. In equation 1, the first part projects each word in the sentence to a real-valued vector, and construct a model as follow:

$$f(w_t, ..., w_{t-n+1}) = \hat{p}(w_t|w_1^{t-1}) \tag{1}$$

where $f(w_t, ..., w_{t-n+1})$ is the trained model that represents the probability $\hat{p}(w_t|w_1^{t-1})$. The second part uses the word vector to construct a probability function instead of the previous function. The raw data input of the model is a vector processed by each word vector, as shown in equation 2:

$$f(w_{t-1}, ..., w_{t-n+1}) = g(C(w_{t-1}), ..., C(w_{t-n+1})) \tag{2}$$

where function C maps sequence of feature vectors to a conditional probability distribution function g. Each word vector X_w computed by the Embedding layer can be expressed as:

$$X_w = {W_e}^{d\times|V|} v^n \tag{3}$$

$$X_{1:L} = [x_1, x_2, x_3, ...x_L] \tag{4}$$

where v is the original input word and W_e is the trained embedding vectors. Containing all Xw, $X_{1:L}$ is the output of the Embedding layer.

One-dimensional convolutional layers are connected behind the Embedding layer. The input to the BiLSTM-prefixed CNN layer is an array of word vectors after Embedding. In the convolutional layer, the filter we used is $v \in \mathbb{R}^{3\times100}$. The filter performs convolution on three word vectors of length 100. We apply 128 filters in convolutional layer with kernel size of 3:

$$f_j^\iota = h(\sum_{i\in M_j} X_{i:i+2}^{\iota-1} v_{i:i+2}^\iota + b_j^\iota) \tag{5}$$

$$F = [f_1, f_2, f_3, ...f_{n-2}] \tag{6}$$

where $X_{i:i+2}$ is embedded word vector and b_j^ι is the bias. The output of each filter is f_i, which is calculated by the filter moving through the set of word vectors. The step size for each move is 1, ensuring that each vector window $\{X_{1:3}, X_{2:4}, ...X_{n-2:n}\}$ can be scanned. F refers to the output of convolution layer.

We perform the BatchNormalization (BN) layer after the 1D convolution layer. BN layer fixes the size structure of F, and solves the gradient problem in the backward propagation process (gradient disappears and explosions) by normalizing activation to a uniform mean and variance, meanwhile, it maintains that different scale parameters should be more consistent in the overall update pace.

F_i is the linear transformation result of the normalize result. The values of γ and β are obtained by the BackPropagation (BP) algorithm.

The Max Pooling layer is connected behind the BN layer. The array after BN goes through a layer of neurons with ReLU activation function.

The output $\tilde{F}$ is a 349×128 two-dimensional array, which is performed by MaxPooling operation.

$$\tilde{F} = MaxPooling\{ReLU(F_1)\} \tag{7}$$

The BiLSTM layer is connected behind the CNN layer. The return sequences parameter is set to True, indicating that the output of each BiLSTM unit is valid and the output will be used as the input to the post-CNN. The BiLSTM layer has a internal structure can be expressed as:

$$c_k^t = i_k^t \circ z_k^t + f_k^t \circ c_k^{t-1}, k \in \{f, b\} \tag{8}$$

where the state of memory cell c_k^t can be affected by the previous state c_k^{t-1} and the input gate i_k^t. o_k^t is the output gate, computed by the input vector x^t and y_k^{t-1}, the output of the previous time step:

$$o_k^t = tanh(W_o^k x^t + R_o^k y_k^{t-1} + b_o^k), k \in \{f, b\} \tag{9}$$

where W_o^k and R_o^k are the weight vectors. y_k^t is the output of BiLSTM layer, of which calculated by o_k^t and the activation funcion (tanh):

$$y_k^t = o_k^t \circ tanh(c_l^t), k \in \{f, b\} \tag{10}$$

At the same time, in order to prevent over-fitting, dropout rate of 0.3 and recurrent dropout rate of 0.3 are added. The output of the BiLSTM layer is a 349×128 two-dimensional array.

The CNN that connected after BiLSTM is similar to the previous CNN layer structure. The number of filters in the convolutional layer is set to 128, the kernel size is 3, and the ReLU activation function is also used. We apply a BN layer before the pooling layer prevents gradient dispersion. The input of CNN is a two-dimensional array of 349×128 and the output is a two-dimensional array of 86×128.

Before accessing the output layer, we set up a Flatten layer to expand the two-dimensional array into a one-dimensional array and a hidden layer containing 64 neurons. An one-dimensional array obtained by Flatten is connected to this layer in a fully connected manner.

The output layer contains only one neuron activited by Sigmoid. Since detecting a malicious request is a binary problem, we chose Binary Crossentropy as the loss function of the model.

The output is a value between 0 and 1. The closer the output value is to 1, the greater the probability that the model will judge the input equest as a malicious attack. Conversely, the closer the value of the output is to 0, the greater the probability that the model will judge the input request as a normal request.

3 EXPERIMENTS AND RESULTS

3.1 Dataset And Training

We evaluate GBLNet using the HTTP data set CSIC 2010. We randomly pick 80% (82416) of the whole dataset as the training dataset, including 57600 normal

request and 24816 exception requests, and 20% (20604, 14400 normal request and 6204 exception requests) as the testing dataset. Each request contains up to 1400 words. For requests with less than 1400 words, we fill it to 1400.

In our experiment, four GTX 1080Ti graphics cards are used for training under the Ubuntu 16.04 operating system. The batch size during training is $64 \times N$ (N is 4, the number of GPU). Meanwhile, we used Keras API to build models based on TensorFlow and train the models for 5 epochs.

3.2 Results and Discussion

We evaluate with 4 commonly used metrics, including Accuracy (Acc), Precision (Pre), Recall (Re) and F1-score (F1).

Table 1. Accuracy, F1-score, Precision and Recall of different models include proposed deep learning methods and improved machine learning methods.

Model	**Acc**	**F1**	**Pre**	**Re**
RNN-IDS [7]	0.6967	0.8210	0.6967	1.0000
HAST-I [11]	0.9886	0.9919	0.9880	0.9958
HAST-II [11]	0.8177	0.8753	0.8301	0.9263
BiLSTM [6]	0.8314	0.8924	0.8083	0.9959
SAE [10]	0.8832	0.8412	0.8029	0.8834
PL-RNN [5]	0.9613	0.9607	0.9441	0.9779
BL-IDS [3]	0.9835	0.9858	0.9900	0.9817
DBN-ALF [1]	0.9657	0.9400	0.9648	0.93200
SVM [8]	0.9512	0.9371	0.9447	0.9296
LR [8]	0.9768	0.9414	0.9236	0.9598
SOM [4]	0.9281	0.7997	0.6977	0.9367
GBLNet	**0.9954**	**0.9967**	**0.9958**	**0.9977**

As shown in Table 1, first, we compare with the deep learning models and the optimized machine learning methods. The accuracy of our proposed model has achieved state of the art (99.54%), which is 29.87% higher than RNN-IDS (69.67%) and 17.77% higher than HAST-II (81.77%). It is also 0.68% higher than that of HAST-I (98.86%). Compared with the optimized machine learning methods, our model performs much better. The accuracy of our method is 6.73% higher than that of SOM, as well as slightly higher than that of SVM (0.95%) and LR (0.97%).

Secondly, we compare the performance among traditional machine learning approaches, including KNN, decision tree, naive bayes and random forest, demonstrated as table 2. Although most traditional machine learning can achieve high accuracy, around 95%, our model is superior to them in all indicators.

Moreover, we also evaluate the models with convergence speed and training speed. Since the dynamic intrusion detection system, as an application type of firewall, needs to defense the malicious attack in real time, and the detection

Table 2. Comparison of proposed model and original machine learning methods.

Model	Acc	Pre	Re	F1
KNN	0.9317	0.9305	0.9760	0.9527
DecisionTree	0.9393	0.9579	0.9559	0.9569
NaiveBayes	0.7432	0.7787	0.8882	0.8298
RandomForest	0.9506	0.9627	0.9673	0.9650
GBLNet	**0.9954**	**0.9967**	**0.9958**	**0.9977**

model should be continuously trained and updated, which emphasizes the cost on convergence speed and training speed should be smaller, the better.

Table 3. Consumed time compared with deep models

Model	Time Consumption
RNN-IDS [7]	14m 22s
HAST-I [11]	37m 78s
HAST-II [11]	7m 9s
BiLSTM [6]	2h 15m
SAE [10]	2h 47m
PL-RNN [5]	28m 17s
BL-RNN [3]	31m 74s
DBN-ALF [1]	1h 27m
GBLNet	**30m 30s**

Table 3 presents the training time of differnt models mainly among the RNN-based and LSTM-Based models. GBLNet costs the least training time among LSTM-based models. It can be seen that the BiLSTM requires more than 2.25 hours to train 5 rounds, while the GBLNet model uses 30m 30s. RNN-IDS, HAST-II reach shorter training time compared with GBLNet, however, RNN-IDS and HAST-II are far worse than our model in terms of accuracy. The results show the advantages of connecting the shadow and deep features maps of the convolutional layers, which plays an important role in speeding up the training by non-linear feature extractors.

4 CONCLUSION

This paper presents a novel strategy to detect malicious requests, and proposes a deep learning model named GBLNet, which girdles the bidirectional LSTM with multi-granularity convolutional features to fully consider the non-linear features of the malicious requests. Applying CNNs before BiLSTM to extract query features successfully maximizes the malicious features of the request queries, leading to much more accurate features representation than that of using BiLSTM to process the queries simply. By connecting the shadow and deep features map

of the convolutional layers, GBLNet can guarantee better feature representations than other temporal models. Evaluations on real-world scenario prove that GBLNet can achieve superior detection rate and faster convergence speed, which promotes the application in the actual dynamic intrusion detection system.

5 Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant U2003111, 61871378).

References

1. Alrawashdeh, K., Purdy, C.: Fast activation function approach for deep learning based online anomaly intrusion detection. In: 2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing,(HPSC) and IEEE International Conference on Intelligent Data and Security (IDS) (2018)
2. Fredj, O.B., Cheikhrouhou, O., Krichen, M., Hamam, H., Derhab, A.: An OWASP top ten driven survey on web application protection methods. In: International Conference on Risks and Security of Internet and Systems (2020)
3. Hao, S., Long, J., Yang, Y.: BL-IDS: Detecting web attacks using bi-lstm model based on deep learning. In: International Conference on Security and Privacy in New Computing Environments (2019)
4. Le, D.C., Zincir-Heywood, A.N., Heywood, M.I.: Unsupervised monitoring of network and service behaviour using self organizing maps. Journal of Cyber Security and Mobility (2019)
5. Liu, H., Lang, B., Liu, M., Yan, H.: CNN and RNN based payload classification methods for attack detection. Knowledge-Based Systems (2019)
6. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing (1997)
7. Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence (2018)
8. Smitha, R., Hareesha, K., Kundapur, P.P.: A machine learning approach for web intrusion detection: Mamls perspective. In: Soft Computing and Signal Processing (2019)
9. Tang, Z., Wang, Q., Li, W., Bao, H., Liu, F., Wang, W.: HSLF: HTTP header sequence based lsh fingerprints for application traffic classification. In: International Conference on Computational Science (2021)
10. Vartouni, A.M., Kashi, S.S., Teshnehlab, M.: An anomaly detection method to detect web attacks using Stacked Auto-Encoder. In: 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS) (2018)
11. Wang, W., Sheng, Y., Wang, J., Zeng, X., Ye, X., Huang, Y., Zhu, M.: HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access (2018)