CNN-Webshell: Malicious Web Shell Detection with Convolutional Neural Network

YifanTian
Army Engineering University
Guanghua Road,
HaifuStreat, No. 1
Nanjing, China
086-15608043621
156080436321@163.com

Jiabao Wang
Army Engineering University
Guanghua Road,
HaifuStreat, No. 1
Nanjing, China
086-18651836993
jiabao_1108@163.com

Zhenji Zhou
Army Engineering University
Guanghua Road,
HaifuStreat, No. 1
Nanjing, China
086-15205158317
zhou_zhenji@163.com

Shengli Zhou
Army Engineering University
Guanghua Road,
HaifuStreat, No. 1
Nanjing, China
086-18705710555
76933768@qq.com

ABSTRACT
Malicious web shell detection is one of the most important methods for protecting the network security. Most state of the art methods are based on malicious keywords matching, where the keywords are usually defined by the domain experts. So its effect depends on the domain experts and it is hard to detect new type of malicious web shells. This paper proposed a new malicious web shell detection approach based on 'word2vec' representation and convolutional neural network (CNN). Firstly, each word, separated from the HTTP requests, is represented as a vector by using the 'word2vec' tool. Next, a web request can be represented as a size-fixed matrix. Finally, a CNN-based model is designed to classify the malicious web shells and the normal ones. Experimental results showed that this approach achieves the best performance, comparing with several other classification methods. To the best of our knowledge, this is the first time that CNN has been applied to malicious web shell detection.

CCS Concepts
•Security and privacy →Artificial immune systems

Keywords
Malicious web shell detection, word2vec, word embedding, convolutional neural network

1. INTRODUCTION
A web shell is a program that is written in web scripting languages, such as ASP, PHP, JSP, CGI, etc. It provides a tool to communicate with the server's operation system (OS) via the interpreter of the web scripting languages[1]. It is always called as webpage backdoor, because a user can use the webpage to upload files, view the database, and execute OS commands through the browser. Hence, malicious user can also start web attacks by the normal web shell with malicious function.

ICNCC 2017, December 8–10, 2017, Kunming, China

ACM ISBN 978-1-4503-5366-3/17/12…$15.00
DOI: https://doi.org/10.1145/3171592.3171593

The common attack process by using web shell has two following stages:

- Using network reconnaissance tools, attackers can identify vulnerabilities that can be exploited and result in the installation of a malicious web shell. For example, these vulnerabilities can exist in content management systems (CMS) or web server software.
- Once successfully installation, attackers can use the web shell to leverage other exploitation techniques to escalate privileges and to issue commands remotely. These commands are directly linked to the privilege and functionality available to the web server and may include the ability to add, delete, and execute files as well as the ability to run shell commands, further executables, or scripts. They can also upload additional malware for the potential of creating a watering hole for infection and scanning of further victims.

Malicious web shells, such as China Chopper, WSO, C99 and B374K, are frequently chosen by hackers. They can be delivered through a number of web application exploits or configuration weaknesses, such as Cross-Site Scripting, SQL injection, exposed admin interfaces. Consistent use of these web shells by Advanced Persistent Threat (APT) and criminal groups has led to significant cyber incidents. Furthermore, these are just a small number of known used malicious web shells. So it is urgent requirement to detect the known and unknown malicious web shell attacking.

Malicious web shell detection methods can be divided into two types: static detection methods and dynamic detection methods [2]. The former have regularization feature matching approach (such as Linux Malware Detect scanner), statistical feature thresholding approach (sush as NeoPI), and association-based detection approach (malicious web shell has lower connections with the existing web files). The latter has the HOOK approach for the key function in web files, and the monitoring approach for OS reading and writing operation in special directories. However, in reality, many malicious web shells are difficult to detect because they are confused with normal webpage files after confusion and mutation. The malicious functionality appears only when the attacker starts the HTTP communication with the target server to execute the malicious codes. So, this paper focus on the malicious HTTP communication to detect malicious web shells. It can find the source client and the target server, and runs in real-time.

Detection of malicious web shells based on the HTTP communication text context can be considered as a special case of

'text' classification. The 'text' is the HTTP request command, which is different from the normal text because it includes many special symbols. Furthermore, it has no space segmentation in the 'text'. If we want to use the common classification techniques in text domain, we have to do 'words' (or symbols) segmentation firstly. Then, the related text classification techniques can be introduced to do this challenge work. The count vectors, TF-IDF vectors, and n-grams are the most used approaches for text feature representation [3,4,5]. To detect whether the HTTP communication contains malicious webshell, machine learning technologies, such as Naive Bayes (NB), k-Nearest Neighbour (kNN), Support Vector Machine (SVM), Decision Tree (DT), and Neural Network (NN), can be used to model the training data, and predict the new HTTP requests[5,6,7,8,9].

Recently, deep learning has been introduced in intrusion detection. Javaid et al. [10] proposed to represent and detect the intrusion by introducing the sparse auto-encoder and soft-max regression. Meanwhile, for word representation in text classification, Mikolov et al. [11,12] proposed the low dimensional word vector representations, 'word2vec'. Based on this 'word2vec' representation, Kim et al. [13] use CNN to classify text. It is a great work for sentence classification. Furthermore, Zhang et al. [14] proposed a character-level CNN for sentence classification. Inspired by the success of the convolutional network in text and image classification, we explored the effectiveness of CNN in malicious web shell detection.

This paper focus on the HTTP request produced in the web service communication and proposed a new malicious web shell detection approach based on 'word2vec' representation and CNN, called as 'CNN-Webshell' for abbreviation. To the best of our knowledge, this is the first time that 'word2vec' and CNN are applied to detect malicious web shells. Our contributions are:

- The 'word2vec' vector representation is introduced to represent the HTTP symbol words, which are segmented from the HTTP requests. It is the first time to represent a HTTP request as a matrix, by concatenating the word vectors in order.
- Once having the matrix representation, CNN can be used to extract features and to learn the classifier simultaneously for detecting the malicious web shell communication. It has the ability of finding the correlation between two words, which are far away from each other.
- Experiments are conducted to evaluate the performance of our proposed approach, which is compared with several classifiers and the key parameters and performance are analyzed.

2. DATA PREPARATION

2.1 Data Collection and Processing

To detect the malicious HTTP requests, we simulated the web shell attack in our intranet. By using the WireShark[1] tool, we collected a set of stream pcap files, with both malicious and normal HTTP requests.This paper focus on the HTTP request content. For POST request, it means the data being posted. For GET request, it means the parameter part of the Request-URI.

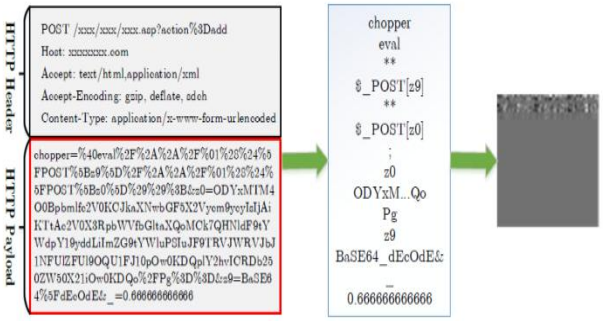

Figure 1. An example of malicious HTTP request, 'word' segmentation and data representation

Fig.1 gives an example of the packet containing the malicious HTTP request. The request content being posted is encoded by URL encoding. So, firstly, we need to decode the parameter string in preprocessing. For each parameter string, it has many parts segmented by the symbol '\&'. Each part consisted of a key-value pair. The key and value in the pair can be obtained by finding the symbol '='. Besides, in practice, a parameter string of the HTTP request also has many other special symbols, such as '(', ')', '{', '}', '/', '\', '@'. These symbols can be used to segment the parameter string into 'words'. This can obtain many meaningful 'words' for classifying the malicious web shells. The segmented 'words' have simple symbols, digits, or normal words and strings. By the end of this step, we get a 'word' sentence for each HTTP request, and the sentence keeps the order of the 'words' in the HTTP request.

2.2 Data Representation

After 'word' segmentation, the primary work is to represent the 'word' sentence. The 'word2vec' tool[2] is used to represent a 'word' as a vector. In practice, the continuous bag-of-words model architecture [11] and the hierarchical softmax [12] are used to represent and compute the transformation network. To illustrate clearly, following is the formulated representation of the process and results:

Given a symbol word *i*, its corresponding vector is $x_i \in R^d$, where *d* is the dimension of the embedding vector space. Given a sentence composed of *n* words, it can be represented as matrix

$$X_{1:n} = [X_1^T, X_2^T, f_i, X_n^T] \in R^{n \times d} \qquad (1)$$

So a sentence of length *n* can be transformed into a matrix of size $n \times d$. With this representation, CNN can be adopted to learn the binary classification problem for malicious and normal web shells.

However, the input of the CNN is expected of the same size in a batch, for speeding the training process. It means that the input sentences must have the same number of words. Suppose the input of the CNN has a fixed sentence length *n*. If a sentence contains less than *n* words, the zero vector is appended to the end of the matrix. If it contained over *n* words, the first *n* words are used to represent the matrix.

[1] https:www.wireshark.org/

[2] https://code.google.com/archive/p/word2vec/

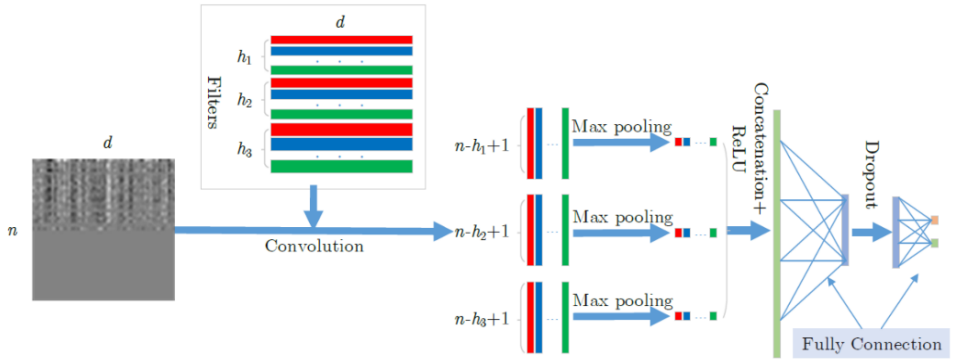

Figure 2. The architecture of CNN-Webshell.

3. CNN-BASED MALICIOUS WEB SHELL DETECTION

Once finished the data representation, a CNN model is designed for classifying malicious and normal web shells. Fig.2 is the architecture of our CNN-webshell. The CNN model is constructed by convolution, pooling, ReLU and fully-connection layers, which are presented as follows:

3.1 Convolution

Given a filter $W \in R^{h \times d}$, where *h* is the filter window width. Here *h* is the number of words for convolution. At position *i* of the given matrix $X_{1:n}$, the result of convolution is

$$c_i = f(W \cdot X_{i:i+h-1} + b) \in R \qquad (2)$$

Where *b* is the bias and *f* is the nonlinear rectify function. The convolution result of the whole sentence is a feature vector

$$c = [c_1, c_2, ..., c_{n-h+1}] \in R^{n-h+1} \qquad (3)$$

In practice, the filter window width *h* can be set to different values, to produce a vector of length *n-h+1*. In experiments, *h* is set as {3, 4, 5}. Besides, for each window width, we use *m* filters to produce *m* vectors. So we can obtain a vector *c(h,m)*, by using the *m*th filter with window width *h*. A total of 300 vectors are produced with *h*=3 and *m*=100.

3.2 Pooling and ReLU

A max-pooling operation on the feature map (result of convolution) is applied to obtain the maximum value *c(h,m)=max{c(h,m)}* as the feature corresponding to the particular filter *w*. It captures the most important value (the highest value) for each feature map. Besides, this operation can naturally deal with variable sentence lengths. Then, the max-pooling results are concatenated to get a feature vector $Z = [z_1, z_2, ..., z_{hm}]$, which is operated by the ReLU activation [15]. ReLU is an element-wise operation, $z_i = max(0, z_i)$. This activation function can get better gradient in back-propagation.

3.3 Dropout and Objective

After that, a fully-connectoin is appended to map the *z* to a hidden layer, with less parameters. Before mapping to the binary output, in training process, a dropout layer is appended. Dropout is an important strategy to suppress overfitting problem [16]. It drops the output weights of each hidden units randomly. In practice, a dropout is used with rate of 0.5.

Finally, the second fully-connection layer is used to map to the binary output, which are normalized to [0,1] by softmax. The bigger one corresponds to the predicted class. The objective of softmax log-loss is

$$L = \frac{1}{N}\sum_{i=1}^{N} -\log \frac{\exp(z_i^{y_i})}{\sum_{k=1}^{C} \exp(z_i^k)} \qquad (4)$$

where $z = z_i$ is the input of softmax log-loss, $z_i^{y_i}$ is the value of z_i at the y_i position.*N* is the number of samples.

4. EXPERIMENTS

4.1 Datasets and Setting

In our experiments, 3691 positive samples (malicious web shells) and 3990 negative samples (normal data) were collected. During training word embedding, no stop-words and no stemming are adopted, to keep the complete information. In experiments, the 'word2vec' embedding dimension is set to 200 as default if specially explanation. The other parameter is set as the default. For CNN training, the length of sentence *n* is set to be 56, and the training epoch is set to be 25 for all the experiments. Batch size for training is also set to 25. The precision, recall, F1-score are used to evaluate the performance. The 10-fold cross-validation is used to evaluate all approaches.

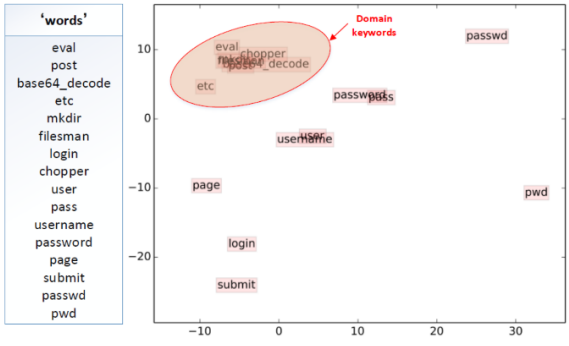

Figure 3. The distribution of 'word' vectors.

4.2 Results and Analysis

Firstly, this paper analysis the effect of the 'word2vec'. The vectors produced by the 'word2vec' can be used to measure the similarity among words. If two words have similar meaning, the two corresponding vectors are close. In the embedding d dimensional vector space, the related words or groups of

words are near to each other. Fig. 3 shows some words, which are embedded into a plane by t-SNE [17] from the vector space. From the figure, we can find that the malicious operation words (the domain keywords in web shell) are very close to each other and the normal words are far from them. So the 'word2vec' vector representation can describe the similarity of the words, and the same labeled words' vector will be similar.

Next, to evaluate the performance of our proposed approach, this paper compared it with several classical classifiers, Naive Bayes (NB), k-Near Neighbour (kNN), Decision Tree (DT), Support Vector Machine (SVM), and traditional Neural Network (NN) using the TF-IDF feature representation. All of these classifiers used the same features. For Multinomial NB, the constant parameter alpha is set as 0.01. For kNN, the neighbor parameter is set as 3. For DT, the entropy is used and the minimal samples in a leaf is set as 3. For SVM, the linear classifier is adopted. For NN, two hidden layers with 50 and 10 hidden units are used. Table 1 shows the Precision, Recall and F1-score performance of each approach in our experiments. The NB approach give the worst performance. Comparing with all other methods, our approach achieves the best performance, due to its good 'words' representation and CNN classifier.

Table 1. Table captions should be placed above the table

	Precision	Recall	F1-Score
NB	0.906	0.891	0.895
KNN	0.940	0.932	0.934
DT	0.949	0.948	0.949
SVM	0.960	0.959	0.960
NN	0.965	0.966	0.965
Ours	**0.986**	**0.986**	**0.986**

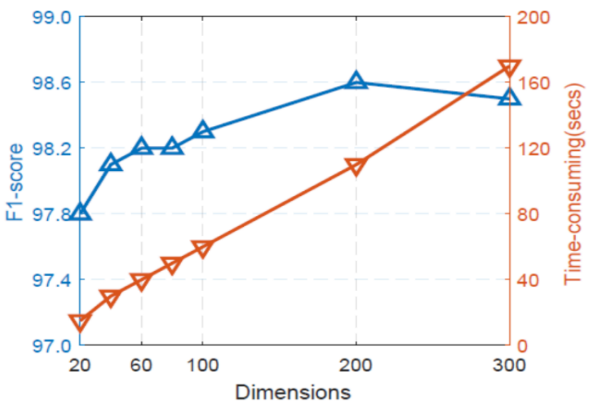

Figure 4. The performance of different dimensions.

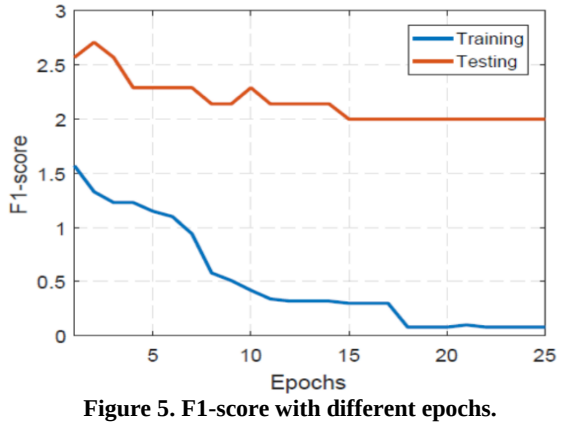

Figure 5. F1-score with different epochs.

As the dimension of the 'word2vec' vector space has significant impact to its representation ability, the paper test the different dimensions to evaluate the performance. Dimension of 20, 60, 100, 200, 300 was adopted. The results are show in Fig. 4, which shows that the best performance is achieved at the dimension of 200.

Besides, the average time-consuming of each epoch was also tested, with different dimensions of the embedding vector space. Fig. 4 shows that the training time-consuming increase as the dimension increasing. So, in practice, the balance between the detection performance and the time-consuming should be taken into consideration.

Finally, the paper presents the convergence for training the CNN model. Fig. 5 gives that curves of the training and validating errors as the epochs. it is obvious that the training process for model of this paper can converge quickly.

5. CONCLUSION

This paper proposed a new malicious web shell detection approach based on 'word2vec' representation and CNN. Words are represented as vectors, which can be represented as a matrix by concatenating them. The proposed CNN model is designed to classify the malicious web shells and the normal ones. Experimental results showed that this approach achieves the best performance, comparing with several classical classifiers. Since the detection of malicious web shells are divided into two stages in this work, learning the vector and classifier simultaneously is put in the further.

6. ACKNOWLEDGMENTS

This work has been supported by the National Natural Science Foundation of China (61402519), and partially supported by the Natural Science Foundation of Jiangsu Province (BK20150721).

7. REFERENCES

[1] Jinsuk Kim, Dong HoonYoo, Heejin Jang, and KimoonJeong. 2015. Webshark 1.0: a benchmark collection for malicious web shell detection. Journal of Information Processing Systems 11, 2 (2015), 229–238.

[2] OleksiiStarov, Johannes Dahse, Syed Sharique Ahmad, Thorsten Holz, and Nick Nikiforakis. 2016. No honor among thieves: a large-scale analysis of malicious web shells. In International Conference on World Wide Web. 1021–1032.

[3] Rung Ching Chen and Su Ping Chen. 2008. Intrusion detection using a hybrid support vector machine based on entropy and TF-IDF. International Journal of Innovative Computing Information & Control 4, 2 (2008), 413–424.

[4] Armand Joulin, Edouard Grave, PiotrBojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. CoRR abs/1607.01759 (2016).

[5] Sergio Pastrana, Carmen Torrano-Gimenez, Than Nguyen Hai, and Agustin Orfila. 2015. Anomalous web payload detection: evaluating the resilience of 1-grams based classifiers. Springer International Publishing. 195–200.

[6] Michal Choras and RafalKozik. 2015. Machine learning techniques applied to detect cyber attacks on web applications. Logic Journal of IGPL 23, 1 (2015), 45–56.

[7] Xiao Yong Lu, Mu Sheng Chen, Jheng Long Wu, Pei Chan Chang, and MengHui Chen. 2017. A novel ensemble decision tree based on under-sampling and clonal selection

for web spam detection. Pattern Analysis & Applications (2017), 1–14.

[8] Emmanuel Michailidis, Sokratis K. Katsikas, and EfstratiosGeorgopoulos. 2008. Intrusion detection using evolutionary neural networks. In Panhellenic Conference on Informatics. 8–12.

[9] Doyen Sahoo, Chenghao Liu, and Steven C. H. Hoi. 2017. Malicious URL detectionusing machine learning: a survey. CoRR abs/1701.07179 (2017).

[10] Ahmad Javaid, QuamarNiyaz, Weiqing Sun, and MansoorAlam. 2016. A deep learning approach for network intrusion detection system. In Eai International Conference on Bio-Inspired Information and Communications Technologies. 21–26.

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeff rey Dean. 2013. Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013).

[12] Tomas Mikolov, IlyaSutskever, Kai Chen, Greg Corrado, and Jeff rey Dean. 2013. Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems 26 (2013), 3111–3119.

[13] Yoon Kim. 2014. Convolutional neural networks for sentence classification. CoRRabs/1408.5882 (2014).

[14] Xiang Zhang, Junbo Zhao, and YannLecun. 2016. Character-level convolutional networks for text classification. Advances in Neural Information Processing Systems 29 (2016), 649–657.

[15] Vinod Nair and Geoff rey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In International Conference on Machine Learning. 807–814.

[16] Geoff rey E Hinton, Nitish Srivastava, Alex Krizhevsky, IlyaSutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. CoRR arXiv:1207.0580 (2012).

[17] Van Der Maaten Laurens, Geoff rey Hinton, and Geoff rey Hinton, Van Der Maaten. 2008. Visualizing data using t-SNE. Journal of Machine Learning Research 9, 2605(2008), 2579–2605.