A Deep Learning Method to Detect Web Attacks Using a Specially Designed CNN

Ming Zhang(✉), Boyi Xu, Shuai Bai, Shuaibing Lu, and Zhechao Lin

National Key Laboratory of Science and Technology on Information System Security, Beijing Institute of System Engineering, Beijing, China
zhangming2013@alumni.sjtu.edu.cn

Abstract. With the increasing information sharing and other activities conducted on the World Wide Web, the Web has become the main venue for attackers to make troubles. The effective methods to detect Web attacks are critical and significant to guarantee the Web security. In recent years, many machine learning methods have been applied to detect Web attacks. We present a deep learning method to detect Web attacks by using a specially designed CNN. The method is based on analyzing the HTTP request packets, to which only some preprocessing is needed whereas the tedious feature extraction is done by the CNN itself. The experimental results on dataset HTTP DATASET CSIC 2010 show that the designed CNN has a good performance and the method achieves satisfactory results in detecting Web attacks, having a high detection rate while keeping a low false alarm rate.

Keywords: Web attacks · Deep learning · CNN

1 Introduction

The Internet has brought great convenience and happiness to people's life. The World Wide Web (abbreviated the Web) is the primary tool for billions of people to interact with the Internet and has made large contributions to the development of the Information Age. However, people are suffering threats and losses increasingly from the Internet. The Cyber-attacks make waves more and more frequently. What is worse, with the increasing information sharing and other activities conducted on the World Wide Web, the Web has become the main venue for attackers to engage in a range of cybercrimes. As early as 2007, Symantec Corporation had observed that instead of trying to penetrate networks with high-volume broadcast attacks, attackers have adopted stealthier, more focused techniques targeting computers through the World Wide Web, and the majority of effective malicious activities have become Web-based [1]. Another security company, Cenzic, reported in 2014 that 96% of the tested internet applications had vulnerabilities with a median of 14 per application, resulting in that hackers are increasingly focusing on and are succeeding with application layer attacks [2]. It is no doubt that Web security deserves enough attention.

There are a number of technical solutions to guarantee the Web security, including Web application security scanners, penetration testing, fuzzing tools used for input testing, Web application firewalls (WAF), Web intrusion detection systems (Web IDS)

D. Liu et al. (Eds.): ICONIP 2017, Part V, LNCS 10638, pp. 828–836, 2017.
https://doi.org/10.1007/978-3-319-70139-4_84

and so on. For Web protection systems, having an effective method that can inspect Web traffic and detect attacks differing from normal behaviors is crucial and fundamental. There are two basic methods to detect Web attacks, the signature-based [3] and the anomaly-based [4]. The signature-based method builds the detection model from known attacks and any behavior having the corresponding attack signatures is identified as an attack. On the contrary, the anomaly-based method creates a profile from normal behaviors and any violation is identified as an attack. Obviously, both of the methods must have enough characterization and generalization ability of abnormal or normal behaviors, whereas it is difficult to do that in practice. With the popularity of machine learning, especially the rise of deep learning, it is possible to let machines learn features and patterns from data and then automatically distinguish different categories of things. In recent years, lots of machine learning methods have been applied to detect Web attacks.

In this paper, we present a method based on deep learning to detect Web attacks, strictly speaking, to detect server-side attacks. We describe a specially designed convolutional neural network and expound the steps and details to detect Web attacks.

The rest of the paper is organized as follows. Some related work is introduced in Sect. 2. The method based on deep learning to detect Web attacks is described in Sect. 3. Experimental results and discussions are presented in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Related Work

Kruegel et al. have presented a multi-model approach to detect Web attacks in [5]. The approach analyzes HTTP requests and uses a number of different models built on different features, including attribute length, attribute character distribution, structural inference, invocation order and so on.

Ma et al. [6] have explored online learning approaches for detecting malicious Web sites using lexical and host-based features of the associated URLs. Their work is to protect users (or clients) from scams, whereas our study is on detecting server-side attacks.

Torrano et al. [7] have proposed an anomaly-based approach to detect intrusions in Web traffic. The approach relies on a XML file to classify the incoming requests as normal or anomalous.

Corona et al. [8] have presented a multiple classifier system to detect Web attacks by modeling legitimate requests. The system employs a set of predefined models, which are established on different message fields in HTTP requests and built on two basic models: the statistical distribution model and the Hidden Markov Model.

Zolotukhin et al. [9] have proposed an anomaly detection method for Web attacks through analysis of HTTP logs. The method employs the n-gram models to extract relevant features from three fields in HTTP logs, including Web resources, query attributes and user agents. Correspondingly, three machine learning algorithms are used, namely, Support Vector Data Description (SVDD), K-means, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

Choras and Kozik [10] have proposed a machine learning approach to model normal behaviors of Web applications and to detect Cyber-attacks. The model is based on information obtained from HTTP requests and consists of patterns that are obtained using graph-based segmentation technique and dynamic programming.

Saxe and Berlin [11] have exposed a deep learning approach to a number of security detection problems including the malicious URLs detection. Similar to our work, they also use the Convolutional Neural Network, but the embedding approaches and the network architectures are different.

3 Method

3.1 Preprocessing HTTP Requests

The HTTP protocol is the foundation of data communication for the Web. The client submits a HTTP request message to the server and the server returns a HTTP response message to the client. If the server is attacked, that means it receives one or more malicious request messages. Based on this, the detection method is designed by inspecting the HTTP request packets (messages) to detect the server-side Web attacks. Because Web attacks detection belongs to the application-layer security solutions and it works after the packets are parsed, our detection method is also available for communications using HTTPS protocol.

An HTTP request message consists of a request line, several request header fields and an optional message body (for POST request messages). Figure 1 is an example of a GET request message obtained from the dataset HTTP DATASET CSIC 2010 (described in Sect. 4).

```
GET http://localhost:8080/iisstart.htm HTTP/1.1
User-Agent: Mozilla/5.0
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml;q=0.9,text/plain;q=0.8
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=11F98280E08EE19274786F4EDDDC821F
Connection: close
```

Fig. 1. A GET request message example.

The request line is the first line of the HTTP request message and comprises three parts: the HTTP-method, the HTTP-url (URL for short) and the HTTP-version. Furthermore, we narrow the detected focus to the URL in an HTTP request message (for POST request messages, the URL is defined as the combination of the HTTP-url and the message body). There are two reasons for doing this. One is that the vast majority of

Web attacks are implemented by manipulating the URLs. The other is for computational convenience. But without loss of generality, our detection method can be applied to the whole HTTP request message.

Suppose a URL is as Fig. 2 shows.

```
http://localhost:8080/tienda1/publico/vaciar.jsp?B2=Vaciar
+carrito%27%3B+DROP+TABLE+usuarios%3B+SELECT+*+FROM+datos+
WHERE+nombre+LIKE
```

Fig. 2. A URL example.

The preprocessing is to segment the URL to a sequence of words (including the string containing some necessary special characters). The URL can be split into words by special characters like "/", "&", "=", "+", etc. If the URL is encoded, the decoding may be operated first. Theoretically, the preprocessing should reflect the difference between abnormal and normal URLs. But in practice, the preprocessing may be a continuously trying and optimizing process. For the above URL, the sequence of words after preprocessing can be as Fig. 3 shows (separated by commas).

```
http, localhost, 8080, tienda1, publico, vaciar.jsp, B2,
Vaciar, carrito, DROP, TABLE, usuarios, SELECT, *, FROM,
datos, WHERE, nombre, LIKE
```

Fig. 3. Sequence of words after preprocessing the URL.

3.2 Word Embedding

Word embedding is used to map the words to vectors, which are the inputs to the Convolutional Neural Network (CNN, described in the next subsection). There are many branches and research groups working on word embeddings. For example, a team at Google led by Tomas Mikolov created *word2vec*, a toolkit that can be used directly to generate vectors. Instead of generated by existing word embedding tools, the words' embedding vectors in our Web attacks detection method are learned in the training process. In fact, there is an embedding layer joint with the CNN, and the embedding vectors are optimized through back-propagations of the whole network. We believe that the embedding vectors generated by this way are more helpful to the detection of Web attacks, because they are task-specific and more reflective of their semantic meaning whereas the vectors generated by third-party tools are relatively static. We have also confirmed our supposition in experiments.

3.3 The Specially Designed CNN

The architecture of the CNN specially designed to detect Web attacks is as Fig. 4 shows. It is a feed-forward neural network and is formed by four distinct layers: the convolution layer, the max-pooling layer, the fully-connected layer and the Softmax layer. The input to the CNN is a matrix composed of vectors embedded from the word sequence which are obtained by preprocessing the URL. Suppose the length of the word sequence is l and the dimension of the embedding vector is k, so the size of the input matrix is $l \times k$. The details of each layer are as follows.

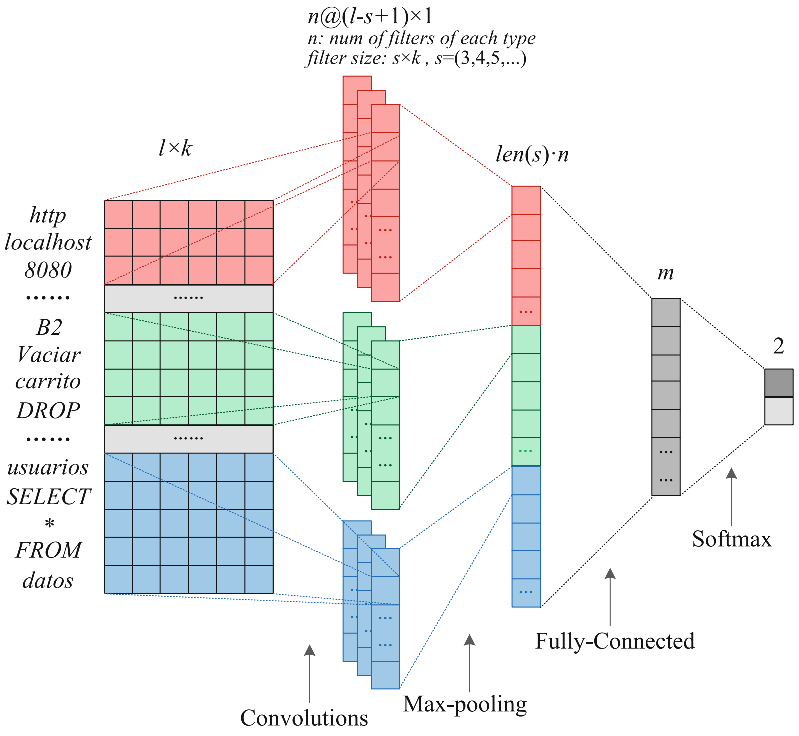


Fig. 4. Architecture of the specially designed CNN.

Convolution Layer

The convolution layer convolves the input matrix with different sizes of filters (or kernels). The filter sizes are set to $s \times k,\ s = (3, 4, 5, \ldots)$, with k equaling to the width of the input matrix and s taking different values. So the types of filters can be represented as $len(s)$. Every filter walks through the input matrix with stride being one and is convolved with a local area of size $s \times k$, thus producing a feature map (the convolved result) of size $(l - s + 1) \times 1$. For each type of filters, there are n filters of the same size to convolve with the input matrix, so the produced results are n feature maps, denoted as $n@(l - s + 1) \times 1$. Because there are $len(s)$ types of filters, the full output of the convolution layer has $len(s)$ clusters actually, with each cluster having n feature maps of size $(l - s + 1) \times 1$. The convolution layer exploits the spatially-local correlation and learns filters that activate when detecting some specific types of features at some local areas of the input. Different sizes of filters can extract more rich features by sliding on different local areas of the input.

Max-Pooling Layer
The max-pooling layer reduces the dimensionality of each feature map outputted by the convolution layer but retains the most important information. It takes the largest element from each feature map, and then concatenates them together to produce a vector. According to the types and numbers of filters in the convolution layer, we can get the length of the max-pooled vector is $len(s) \cdot n$. Note that though the inputs may have different lengths, but the results of the max-pooling layer are always of the same length.

Fully-Connected Layer
The fully-connected layer has m $(m < len(\mathrm{s}) \cdot n)$ neurons, with each neuron connected to all the neurons in the max-pooling layer. The fully-connected layer does high-level reasoning and extracts high-order features of the input. Because $m < len(\mathrm{s}) \cdot n$, it also has the function of reducing dimensionality.

Softmax Layer
The CNN designed for detecting Web attacks is to recognize whether the HTTP request message is normal or abnormal, so the Softmax layer (also the output layer) has 2 neurons with each representing normal or abnormal. The neurons in the Softmax layer are also fully connected to the neurons in the previous fully-connected layer. The output values of the 2 neurons are set to lie between 0 and 1 and sum to 1. The predicted class of the URL is decided by which neuron outputs a larger value.

Some other tricks may be considered. For example, one can add dropout after the max-pooling layer or other layers to avoid overfitting at the training stage. The choice of the loss function is also critical. For our designed CNN, the cross-entropy loss is recommended.

4 Experiments and Results

To evaluate the effectiveness of the method for detecting Web attacks, we conducted experiments on the dataset HTTP DATASET CSIC 2010 [12].

4.1 Data Preparation

The HTTP DATASET CSIC 2010 dataset contains thousands of Web requests automatically generated by the Information Security Institute of CSIC (Spanish Research National Council), and has been widely used for testing the Web attacks detection systems. The dataset contains 36,000 normal requests and 24,668 abnormal requests. The abnormal requests include Web attacks such as SQL injection, buffer overflow, information gathering, files disclosure, CRLF injection, XSS, server side include, parameter tampering and so on.

We randomly selected about 70% of the dataset as training data, 5% as the validation data, the rest 25% as the test data. The data distribution is shown in Table 1.

Table 1. Experimental data distribution.

	Training	Validation	Test
Normal	25,200	1,800	9,000
Abnormal	17,268	1,233	6,167
Total	42,468	3,033	15,167

4.2 Model Parameters and Evaluating Criteria

Based on data characteristics and empirical experiences, we set parameters of the CNN as follows. For the input matrix, the height l depends on the length of the word sequence, and the width k, i.e. the dimension of the embedding vector is set to 128. In the convolution layer, for the filter size $s \times k$, let $s = (3,\ 4,\ 5,\ 6)$ and $k = 128$, so $len(s) = 4$. For each type of filters, the number n is set to 128. The neurons number m of the fully-connected layer is set to 256. We add dropout after the max-pooling layer at the training stage with the keeping probability being 0.5 and choose cross-entropy as the loss function.

To evaluate the effectiveness of the method for detecting Web attacks, we use three criteria: the *detection rate*, the *false alarm rate* and the *accuracy*. Following the notions usually used in machine learning methods, we use *TP*, *FP*, *TN* and *FN* to represent the number of true positives, false positives, true negatives and false negatives respectively. The *detection rate* (i.e. *true positive rate*) is defined as the proportion of the detected abnormal requests accounting for the total abnormal ones. The *false alarm rate* (i.e. *false positive rate*) is defined as the ratio between the number of normal requests wrongly categorized as abnormal and the total number of actual normal requests. An ideal Web attacks detection method must have both the high *detection rate* and the low *false alarm rate*. The *accuracy* is defined as the proportion of requests including normal and abnormal to be correctly classified. The formulas of the three criteria are presented below.

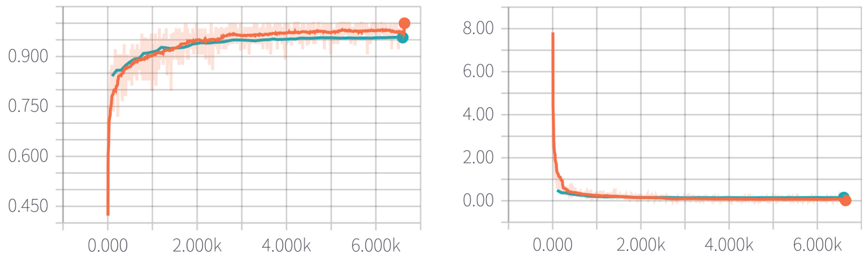
$$detection\ rate = \frac{TP}{TP + FN}. \tag{1}$$

$$false\ alarm\ rate = \frac{FP}{FP + TN}. \tag{2}$$

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{3}$$

4.3 Results and Discussions

We trained the CNN for 10 epochs using batch training approach. The batch size is set as 64. We recorded the training accuracy and loss every one step and recorded the validation accuracy and loss every 100 steps. The trends of the metrics are presented in Fig. 5. Figure 5(a) shows the accuracy trends, where the orange curve represents the training accuracy and the dark cyan represents the validation accuracy. We can see that after about 4,000 steps (about 6 epochs) of training, both the training and validation accuracies have achieved above 95%. Figure 5(b) shows the loss trends, where the orange curve represents the training loss and the dark cyan represents the validation loss. Obviously, both the training and validation losses decrease rapidly towards 0. Such trends of accuracy and loss reflect the good performance of the CNN.



(a) accuracy (b) loss

Fig. 5. Accuracy and loss in the training stage.

After 10 epochs of training, we run the trained CNN on test data to evaluate its ability of detecting Web attacks. As Table 2 shows, the *detection rate* is 93.35%, the *false alarm rate* is 1.37%, and the *test accuracy* is 96.49%. This demonstrates that with a certain amount of training, the CNN has achieved satisfactory results in detecting Web attacks, having a high detection rate while keeping a low false alarm rate.

Table 2. Evaluating results.

Detection rate	*False alarm rate*	*Test accuracy*
93.35%	1.37%	96.49%

5 Conclusion

A deep learning method to detect Web attacks is explored, which is based on a specially designed convolutional neural network. The method is able to detect various Web attacks through inspecting the HTTP request packets. Firstly, data preprocessing is studied, which chooses useful information from HTTP request packets and produce lots of word sequences. Secondly, the embedding approach used to map words to vectors is studied. The embedding vectors are learned in the training stage and not generated by the existing word embedding tools. Finally, a special CNN consisted of various layers is designed. It is able to extract features automatically, and then classify the HTTP request packets to normal or abnormal class. We conducted experiments on the dataset HTTP DATASET CSIC 2010 to evaluate the effectiveness of the method. The results show that the designed CNN can be trained easily and the detection method achieves a high detection rate with few false alarms in detecting Web attacks.

For reducing computational complexity, the method in this article only focuses on detecting Web attacks hidden in URLs. Future work will try modeling the whole HTTP request messages. Other embedding approaches and neural networks are also worth studying.

References

1. Symantec Internet Security Threat Report: Trends for July–December 2007. http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_exec_summary_internet_security_threat_report_xiii_04-2008.en-us.pdf
2. Application Vulnerability Trends Report 2014. http://www.cenzic.com/downloads/Cenzic_Vulnerability_Report_2014.pdf
3. Axelsson, S.: Research in intrusion-detection systems: a survey. Technical report 98–17, Department of Computer Engineering, Chalmers University of Technology (1998)
4. Garcia, T.P., Diaz, V.J., Macia, F.G., et al.: Anomaly-based network intrusion detection: techniques, systems and challenges. Comput. Secur. **28**(1), 18–28 (2009)
5. Kruegel, C., Vigna, G., Robertson, W.: A multi-model approach to the detection of web-based attacks. Comput. Netw. **48**(5), 717–738 (2005)
6. Ma, J., Saul, L.K., Savage, S., et al.: Identifying suspicious URLs: an application of large-scale online learning. In: Proceedings of 26th Annual International Conference on Machine Learning, pp. 681–688 (2009)
7. Torrano, G.Z., Perez, V.A., Maranon, G.A.: An anomaly-based approach for intrusion detection in web traffic. J. Inf. Assur. Secur. **5**(4), 446–454 (2010)
8. Corona, I., Tronci, R., Giacinto, G.: SuStorID: a multiple classifier system for the protection of web services. In: Proceedings of IEEE 21st International Conference on Pattern Recognition (ICPR), pp. 2375–2378 (2012)
9. Zolotukhin, M., Hamalainen, T., Kokkonen, T., et al.: Analysis of http requests for anomaly detection of web attacks. In: Proceedings of IEEE 12th International Conference on Dependable, Autonomic and Secure Computing (DASC), pp. 406–411 (2014)
10. Choras, M., Kozik, R.: Machine learning techniques applied to detect cyber attacks on web applications. Log. J. IGPL **23**(1), 45–56 (2015)
11. Saxe, J., Berlin, K.: eXpose: a character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. arXiv preprint arXiv:1702.08568 (2017)
12. HTTP DATASET CSIC 2010. http://www.isi.csic.es/dataset/