2018 IEEE 14th International Colloquium on Signal Processing & its Applications (CSPA 2018), 9 -10 March 2018, Penang, Malaysia

Web Application Firewall using Character-level Convolutional Neural Network

Michiaki Ito Hitoshi Iyatomi
Applied Informatics
Faculty of Science and Engineering
Hosei University, Japan
Email: {michiaki.ito.3a@stu., iyatomi@}hosei.ac.jp

***Abstract*—Web applications can be maliciously exploited by malicious HTTP requests. Normally, web application firewall (WAF) protects web applications from known attacks using pattern matching method. However, introduction of WAF is usually expensive as it requires the definition of patterns according to the situation. Furthermore, the system cannot block unknown malicious request. In this paper, we come up with an efficient machine learning approach to solve these issues. Our approach uses Character-level convolutional neural network (CLCNN) with very large global max-pooling for extracting the feature of HTTP request and identify it into normal or malicious request. We evaluated our system on HTTP DATASET CSIC 2010 dataset and achieved 98.8% of accuracy under 10-fold cross validation and the average processing time per request was 2.35ms.**

***Index Terms*—deep learning; web application firewall; network intrusion detection**

I. Introduction

Usual firewall blocks malicious packets with information up to the transport layer in TCP/IP model, whereas web application firewall (WAF) [1] does with that in the application layer. Therefore, WAF has a capability to deal with attacks on the application layer that ordinary firewall does not. Typical WAF blocks malicious or suspicious packets with pattern matching strategy referring to pre-defined signatures. The strategy can be categorized into two; the blacklist-based and the whitelist-based methods. The former blocks packets by referring pre-defined signatures of malicious ones, while the latter allows only transactions with those of normal traffic patterns. The blacklist-based methods cost relatively low because what one need is to define “black” signatures based on known attacks. Instead, they usually cannot treat unknown and subspecies attacks. By contrast, the whitelist-based methods are likely to block those attacks, while they need the detail definition of “white” signatures based on normal transactions on the application to be protected; thus they are usually expensive. Also, since they do not allow any transactions other than on the list, their applicable fields are limited.

In such backgrounds, several studies have been proposed to reduce introduction costs and improve detection accuracy of malicious packets using machine learning methods [2-6].

Wakiou et al. [2] detected complicated SQL injection attacks using naïve Bayes method and hybrid pattern matching system. Their system showed 97.6% in precision. Zhang et al. [3] detected attacks within HTTP traffic using naïve Bayes and achieved an accuracy of 82.3%. Although these methods achieved certain degree of detection accuracy for attacks, there is a room for improvement in terms of practicality.

Meanwhile, in the field of machine learning, deep learning techniques have been gaining a great deal of results and attracting attention. In conventional machine learning scheme, the fighting with two main problems; difficulty in design and implementation of efficient hand-made features for classification and fighting with over-fitting was continuing painful development.

Convolutional neural networks (CNN) [7] is a representative machine learning method in deep learning, mainly used for image recognition problems. CNN has overcome such difficulties thanks to their convolutional architecture and well-considered regularization methods to suppress the over-fitting and show representative research achievements in wide applications, especially in image recognition [8, 9].

In natural language processing（NLP）task, recurrent neural network (RNN) [10] and long short term memory (LSTM) [11] are often used and demonstrated their effects [12]. In particular, LSTM is widely used in recent years because time series data can be processed while considering past information longer than RNN. However, training cost of LSTM is generally expensive. Character-level CNN (CLCNN), as the name imply is a CNN specialized in text processing and it convolves character strings in a one-dimensional direction. CLCNN is also frequently used in NLP task recently and shows promising results [13]. The advantage of CLCNN is that the training time is much less time than that of LSTM [14], and it has high affinity for many effective and well-known strategies derived from CNN. They are for example, data augmentation on input dimension, drop-out for regularization, analyzing methodologies of feature maps for model interpretation, and so on.

In security studies with deep learning technique, Melicher et al. [4] measured the ease of password guessing using LSTM and achieved up to 70% in accuracy, that is better than Markov model, probabilistic context-free grammar (PCFG) methods.

Raff et al. [5] analyzed whole Windows portable executable (PE) file as a character string for malware identification with CLCNN and achieved up to 94% in accuracy. They suggested that investigating whole string sequence is necessary and the effectiveness of global max-pooling in the cases where mali-

978-1-5386-0388-8

cious parts are unevenly distributed in a short part among the target to be analyzed. Saxe et al. [6] created a malicious URLs discriminator using CLCNN with different sizes (character chunk lengths are between 2 to 5). In their model, the number of input characters is limited to 200 and if the input does not fit, they used only the backward. The number of convolution layer is only one as the same as [5] and, the feature maps convolved with different sized kernels are combined and then analyzes it via the three full connection layers. They reported the AUC (area under the ROC curve) of 0.993, which is about 0.008 higher in AUC than the N-gram + Deep neural network methods on the same data.

From the above research, the method such as CLCNN is effective for analyzing data in the security field. Although their study was quite well organized and presents promising performance, the following points are concerned when we face large-scale data to handle; (1) Since the model has different convolution sizes in parallel, it is impossible to fully exploit the parallelism in the single GPU to cause delay, and (2) large GPU memory usage due to three relatively large fully connected (FC) layers.

In this research, we constructed an identification system for malicious HTTP request assuming practical WAF implementation. Our system is composed of simple CLCNN architecture receives the full text of the HTTP request as the input and is characterized by the fact that performing a very large global max-pooling. Large max-pooling significantly reduces the dimension of input even in shallow network architecture, realizing high accuracy while suppressing necessary resources.

II. Method

A. Dataset

In this study, we used HTTP DATASET CSIC 2010 dataset [15]. This dataset was produced by the Spanish Research National Council (CSIC). It is a summary of the traffic on the Spanish e-commerce web application, which includes 36,000 normal traffic and over 25,000 malicious traffic.

Among malicious traffic, there are attack HTTP requests such as SQL injection [16], buffer overflow attack [13], information gathering. An example of SQL injection is shown in Fig.1. In this example, the highlighted part is a character string related to attack. The minimum, average, and maximum length of the query was 473, 583, and 983 characters, respectively.

B. Preprocess for dataset

Since the raw dataset contains raw HTTP requests, URL encoding has already been done for all characters within the request. When this encode is fed to the model, it is difficult to extract useful features of queries because a large amount of escape characters of '%' are included. Therefore, we performed URL decode for all queries, and encoding again with Unicode; each character included in the character string of the HTTP request is represented by an 8-bit numeric string.

We determined the input request length of CLCNN was 1000 characters based on the distribution of the length of each request and the findings suggested by Raff et al. We

```
GET http://localhost:8080/tienda1/publico/anadir.jsp?id=2&nombre=Jam%F3
n+lb%E9rico&precio=85&cantidad=%27%3B+DROP+TABLE+usuarios%3B+
SELECT+*+FROM+datos+WHERE+nombre+LIKE+%27%25&B1=A%F1adir+al+carrito HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
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=B92A8B48B9008CD29F622A994E0F650D
Connection: close
```

Fig. 1. Example of one SQL injection request data in the HTTP DATASET CSIC 2010

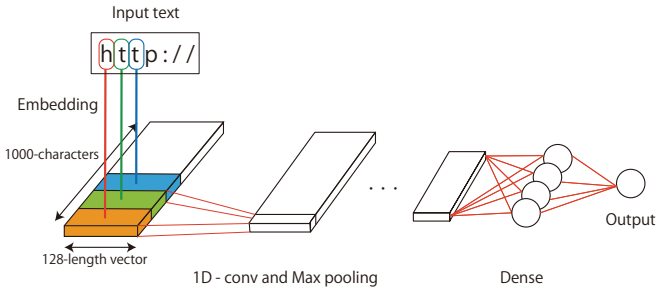

Fig. 2. Outline of our malicious query detection system using CLCNN

padded with '0' representing null in Unicode for the portion insufficient in length.

C. Malicious query detection system using CLCNN

Fig. 2. shows the outline of our malicious query detection system. The system has the embedding layer at first which performs preprocessing and embedding as described in the previous section. Each letter of the input HTTP request sentence is converted into a 128-dimensional vector expression in the Embedding layer, and propagated to the subsequent layers.

We defined the basic structure, the architecture A, determined based on preliminary experiments. The configuration of this is shown in Fig. 3.

The architecture A performs convolution and max-pooling twice and is connected to one-dimensional output via the full connect layer. In the second max-pooling, the pooling size is set as the same as the output dimension of the convolution, i.e. convolved each map yields 1×1 output with this process and accordingly they produce the vector whose dimension matches the number of immediate feature maps. Hyper parameters of architecture A is only one; kernel size K. In all convolution layer, $1 \times K$ kernel is applied and this parameter is also used in the first global max-pooling. The convolution and the stride size of pooling were all fixed at 1. All the activation functions use the rectified linear unit (ReLU) function.

Again, our architecture is characterized by extremely large dimensional reduction at the second max-pooling (e.g. 498: 1

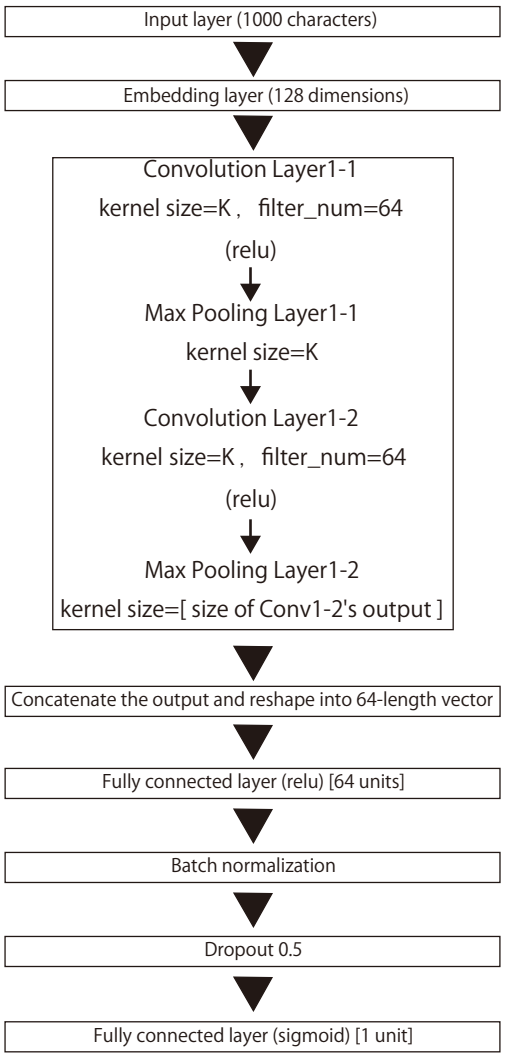

Fig. 3. The structure of architecture A determined based on preliminary experiments

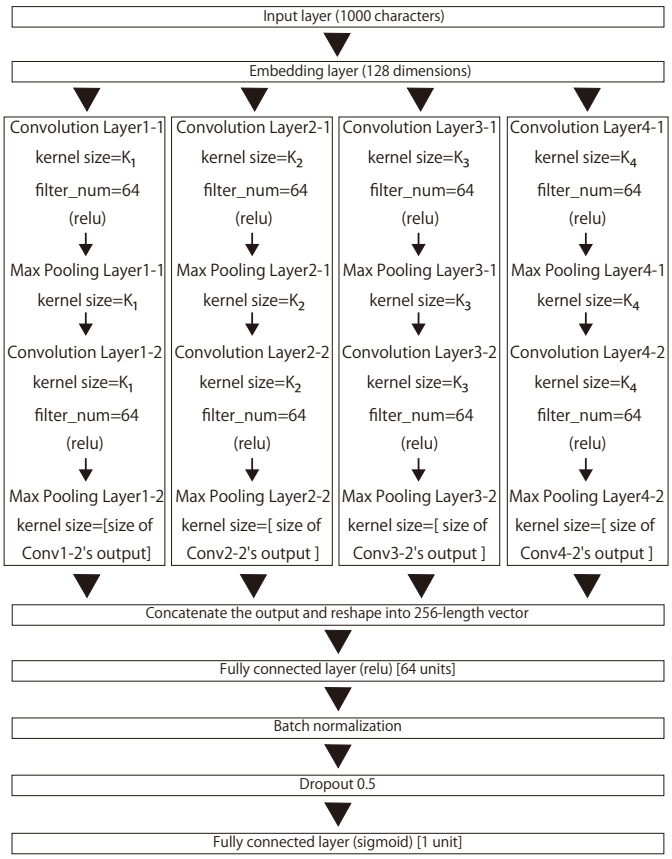

Fig. 4. Structure of architecture B referring to Saxe et al. [5]

TABLE I
RESULT OF THE EXPERIMENT USING ARCHITECTURE A

Kernel size K	2	3	4	5	6	**7**	8
Accuracy [%]	97.5	97.6	98.1	98.3	98.7	**98.8**	97.8

at $K = 2$), so the number of FC elements in subsequent layer is greatly reduced. This greatly reduces the size of the GPU memory required.

For comparison, we built the architecture B as shown in Fig. 4 which concatenates four architectures as in parallel, referring to Saxe et al. [5] and finally merges them. The hyper parameters of architecture B are the size of the convolution kernel size alike the architecture A, but have four; K_1, K_2, K_3, K_4. They are also applied in the size of the first max-pooling. The stride sizes of convolution and max-pooling are all 1.

The number of feature maps in each convolutional layer of architecture B equal to $1/4$ of architecture A, that is, the total number of them in architecture B is the same as architectures A.

III. EXPERIMENTAL RESULTS AND ANALYSIS

In the experiment using architecture A, we compare the discrimination ability when changing the chunk size cut out from the HTTP request character string. Experiment B is aimed at verifying the effect of combining different convolution sizes, in which good results have been obtained in the past research, in a situation where the other conditions are the same. Each evaluation was done with 10-fold cross validation. Results obtained using architecture A and B are shown in Tables 1 and 2, respectively. Each result represents the average value of five trials.

The highest classification accuracy was achieved when K = 7 in architecture A and (K_1, K_2, K_3, K_4) = (4, 5, 6, 7) in architecture B. In above condition, the average execution time including preprocessing by architecture A and B was $2.35ms$ and $3.40ms$, respectively. These experiments used a single GPU (Titan X pascal).

From these results, the simple architecture A showed supe-

TABLE II
RESULT OF THE EXPERIMENT USING ARCHITECTURE B

Kernel size (K_1, K_2, K_3, K_4)	(2, 3, 4, 5)	(3, 4, 5, 6)	**(4, 5, 6, 7)**	(5, 6, 7, 8)
Accuracy [%]	95.2	98.0	**98.2**	97.8

TABLE III
THE RESULT ON HTTP DATASET CSIC 2010

Method	accuracy [%]	time [ms]
our CLCNN (architecture A)	**98.8**	**2.35**
our CLCNN (architecture B)*	98.2	3.40
Zhang et al.[3]	83.2	n/a

*:referring to Saxe et al.[5]

rior performance to architecture B both in the execution speed and the accuracy.

IV. DISCUSSION

Our methods achieved more than 10% better accuracy than Zhang's method [3] using the same dataset even though it is composed of a very shallow network structure. Among all our tested shallow models, the K = 7 configuration of architecture A showed the best results. The ratio of global max-pooling with K = 7 was about 136:1, in fact it carried out extremely large information compression. In its extreme, at K = 2, i.e. the ratio of global max-pooling was 498:1, our model still achieved very high detection accuracy of 97.5%. This is probably because it is possible to equivalently search long strings by convolution and large pooling. We also examined the configuration of the deeper convolution layer and the global average pooling, but they showed lower detection capability. These results support the fact that large global max-pooling is effective when malicious strings can be in the whole part as suggested in literature [6].

Architecture A of a simple configuration realized equal or higher accuracy than the architecture B constructed with reference to Saxe et al [5], and the execution time of architecture A was shorter than architecture B. Here, our model deals with all character strings of http request as input and thus the input size of our model is getting bigger. In such situations, we may think applying deeper network structures with successive convolution is preferable as is common in image recognition. However, large dimensional reduction at global max-pooling in our model attained the smaller dimension of FC layer than that in literature (64×64 for $1024 \times 1024 \times 1024$ in [5]) with shallow structures. For this reason, the superiority at the execution speed mentioned above is not only fast, but also from the viewpoint of the more important discrimination ability, over-fitting is more difficult to occur. By the way, we confirmed some wrong data is included the dataset, some HTTP requests which labelled as malicious has no malicious character strings. So, the accuracy of the dataset used in this experiment is considered to be almost an upper limit.

V. CONCLUSION

We proposed a system to realize fast, accurate and low-cost WAF system by using deep learning approach. Our system achieves these desirable features by adopting a shallow CLCNN configuration that performs extremely large global max-pooling. We will evaluate our system in practical situation in near future.

REFERENCES

[1] A. Endraca, B. King, G. Nodalo, M. S. Maria and I. Sabas, "Web Application Firewall (WAF)," *International Journal of e-Education, e-Business, e-Management and e-Learning*, vol. 3, no. 6, pp. 451-455, December 2013.

[2] A. Makiou, Y. Begriche, and A. Serhrouchni, "Improving Web Application Firewalls to detect advanced SQL injection attacks," *in Information Assurance and Security (IAS), 2014 10th International Conference on*, pp. 35-40, 2014.

[3] Z. Zhang, R. George, and K. Shujaee, "Efficient detection of anomolous HTTP payloads in networks," *SoutheastCon*, pp. 1–3, 30 March-3 April 2016.

[4] W. Melicher, B. Ur, S. M. Segreti, S. Komanduri, L. Bauer, N. Christin and L. F. Cranor, "Fast, Lean, and Accurate: Modeling Password Guessability Using Neural Networks," *26th USENIX Security Symposium*, August 10-12 2016.

[5] J. Saxe and K. Berlin, "eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys," *CoRR*, abs/1702.08568, 2017.

[6] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro and C. Nicholas, "Malware Detection by Eating a Whole EXE," *CoRR*, abs/1710.09435, 2017.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *IEEE proc.*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[8] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, abs/1409.1556, 2014.

[10] Y. Bengio, J. Louradour, R. Collobert and J. Weston, "Curriculum Learning," *Proc. of the 26th Annual International Conference on Machine Learning*, pp. 41–48, 2009.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] R. Johnson and T. Zhang, "Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings," *Proc. of the 33rd International Conference on Machine Learning*, vol. 48, pp. 526–534, June 19-24 2016.

[13] D. Shimada, R. Kotani, and H. Iyatomi, "Document classification through image-based character embedding and wildcard training," *IEEE Proc. Big Data*, pp.3922–3927, 2016.

[14] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-Recurrent Neural Networks," *CoRR* arXiv: 1611.01576, 2016.

[15] HTTP DATASET CSIC 2010. [Online]. Available: http://www.isi.csic.es/dataset. [Accessed: 27-Nov-2017].

[16] S. Gupta, "Buffer Overflow Attack," *IOSR Journal of Computer Engineering*, vol. 1, no. 1, pp. 10–23, May-June 2012.