ASCII Embedding: An Efficient Deep Learning Method for Web Attacks Detection

Ines Jemal1(✉), Mohamed Amine Haddar2,3, Omar Cheikhrouhou1,3, and Adel Mahfoudhi1,3

1 ENIS, CES, LR11ES49, University of Sfax, 3038, Sfax, Tunisia
ines.jemal@studenis.tn
2 ReDCAD Laboratory, University of Sfax, Sfax, Tunisia
3 College of Computer and Information Technology, Taif University, Taif, Saudi Arabia

Abstract. Web security is a homogeneous mixture of modern machinery and software technologies designed to protect the personal and confidential data of all Internet users. After many decades of hard work in web security, the protection of personal data remains an obsession for legitimate internet users. Nowadays, artificial intelligence techniques are overcoming classical signature-based and anomaly-based techniques, which unable to detect zero-day attacks. To help reduce fraud and electronic theft at the server-side, we propose in this paper a novel deep learning method to preprocess the input of neural networks. This technique, called ASCII Embedding, aims to efficiently detect web server attacks. Using an online real dataset CSIC 2010, we evaluated and compared our technique to existing ones as word, and character embedding approaches. The experimental results prove that our technique outperforms the existing works accuracy and reaches 98.2434%.

Keywords: Web security · Web attacks · Convolutional neural network · Word embedding · Character embedding · ASCII code

1 Introduction

The World Wide Web has made our daily life easier. We can buy our needs, pay our bills and consult the news using the Internet. This technology is responsible for electronic fraud. Web Attacks persist even with the more sophisticated security policies and standards. Skilled hackers can overtake security controls and steals sensitive data. To enhance the web application security, the cyber world endows several prevention software tools ready-made such as Web Application Firewalls (WAF) [1], Intrusion Detection Systems (IDS) [2]. The majority of these products use anomaly based and signature-based techniques. These techniques that are developed to secure the server, are unable to stop zero-day attacks. Nowadays, Artificial Intelligence (AI) [3] shows a significant aptness in many domains.
and particularly web security. Among the very popular techniques of AI comes the machine learning [4]. The key concept of this technique is making the machine learns from the existing data. After a while, the machine becomes able to get the right decision. The learning phase is based on a huge amount of data that helps the machine learns how to react to each existing situation. After this phase, the machine makes for new situations a decision with an accuracy concurrent to the human one. The more adequate technique in machine learning is the Neural Network. This technique mimics the human brain and makes somehow the machine reacts intelligently to unknown circumstances. For web security purposes, the neural network learns from a huge number of Http requests stored in a chosen dataset. To reach a higher attack detection accuracy rate, the neural network input data must be carefully preprocessed from the rough dataset requests. The most used techniques in the literature to process the Http request are the word embedding [5] and the character embedding [6]. The first transform word by word the Http request to a vector of numbers. The second approach does the same task at the character level. However, the main problem of these techniques is the unpredictable behavior of the neural network for new characters or words.

To tackle this problem, in this paper, we propose a new technique of Http request preprocessing called ASCII Embedding. This technique outperforms existing techniques by increasing neural network performances. A set of experiments are conducted to measure the performance of the new method. Our technique proved experimentally that it performs better than the existing approaches, it fulfills 98.2434% of accuracy.

The remainder of this paper is organized as follows: The second section presents the state of the art of neural networks preprocessing techniques. Our method is presented in Sect. 3. The evaluation of our method together with the existing techniques is conducted via an experimental test and exposed in Sect. 4. This section lays the results and compares our technique to the existing ones. The last section concludes this paper and briefly explains our future work.

2 Related Works

Studying machine learning and especially deep learning techniques have realized enormous attraction after the success proved in the image processing domain. To get benefit from this success, many domains have applied these techniques such as web security. To detect the web attacks, more strictly, detect the attacks at the server-side, many works proposed the use of deep learning techniques. Mickiaki et al. [7] investigated on Character Level Convolution Neural Network (CLCNN) to detect web application attacks. Each character included in the Http request was represented by 8 bits numeric string and converted into 128-dimensional vector expressions. The authors used the CSIC 2010 [8] dataset to train their model and test their approach. They compared two different architecture using different kernel sizes. The accuracy varied between 95% and 98%. The same dataset was used by Zhang et al. [9], they dissected the Http request into words and deleted the non-alphanumeric characters. Their CNN model achieved 96.49% of accuracy. The authors believed that the embedding vectors generated by word embedding
approach are more helpful to detect web attacks, while Hung et al. [10] considered that using word embedding has some limits because the HTTP request contains too many unique words that deal with memory constraints, and in many cases, it cannot obtain the embedding vector for each new words at the test time. For this reason, Hung et al. [10] proposed an advanced word embedding approach to solving the problem of many words observed. They applied the CNN to both characters and words approaches to process the URL String. The authors used the AUC ROC curve [11] to show the performance of their approach, it achieved 98%. Jane et al. [12] used two neural networks multilayer Feed Forward networks, the first neural network checking by trained web application data, while the second neural network checking by trained the behavior user. The input of the first neural network (ANN1) is defined by a set of collected information from several sections of HTTP request: Post Parameters, Get Parameters, Cookie-Parameters, database operations, file system operations, errors, warning. The second neural network (ANN2) stores data about user behavior as an input: address IP, country, name, version of the browser, version of the operating system, language system, screen resolution, color depth, and browser home page. The two neural networks ANN1 and ANN2 achieved 92% and 95% of accuracy, respectively. Joshua et al. [13] used the eXpose neural network based on character embedding approach to detect malicious URL, they investigated the automatic extraction of features for a short string. Their model achieved a 92% detection rate.

Ernxue et al. [14] used CNN to improve IDS systems. The authors used the word embedding approach to detect malicious payloads, whereas our study is on detecting attacks on the server-side. Their model achieved 95.75% of accuracy.

The previously presented work has used either the word embedding or the character embedding approaches to process the HTTP requests. Trying to enhance the attack detection rate, we propose a new embedding approach called ASCII Embedding, which is presented in the next section.

3 ASCII Embedding Method

The works presented in the literature review use word or character embedding techniques to HTTP request preprocessing. These two approaches are widely used for text processing. The word embedding approach starts by cleaning the HTTP request from the non-alphanumeric symbols. Then, it maps each word to a vector of real values. The character embedding is an interpolation of word embedding to the character level. The works that used these techniques and especially word embedding show acceptable accuracy. Nevertheless, when the HTTP request contains too many words or new words, the decision made is very likely to be incorrect.

We present in this section our new method called ASCII Embedding. The new approach is an interpolation of character embedding to integers. A replacement of characters by their machine value code (like ASCII) is considered. The idea of using ASCII code comes from the successful results showed when using the CNN models for image classification, where the features engineering are the pixels. Every pixel has a value between 0 and 255. The HTTP request is then transformed into a sequence of numbers. Each number represents the ASCII code of the
corresponding character. We choose the Convolutional neural network to test our technique because it has a special architecture and it can learn important features from a big scale. Our new preprocessing method aims to help the neural network to detect efficiently the malicious HTTP requests even with new words not learned during the training phase. In our case, we are interested in the request sent from the client computer and received by the webserver. Figure 1 describes the different parts of the HTTP request. Each part contains eventually a set of words, characters, and symbols.

To not lose any information and to ensure a high detection rate, our method is applied to the whole HTTP request without deleting any word, character, or even symbol. The preprocessing of the HTTP request consists of two successive steps. First, we split the HTTP request into a sequence of characters and symbols. Then, we replace each character or symbol with its integer value (the ASCII code). The result of these steps is a vector $V$ of integer (see Fig. 2).

The output $V$ of the HTTP request preprocessing is the input of the CNN.

---

**Fig. 1.** A real example of HTTP request

```
GET http://localhost:8080/tienda1/publico/productos.jsp 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+xml,html/text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: gzip, deflate, gzip, deflate
Accept-Charset: utf-8, utf-7;q=0.5,*;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=6075D9A19F57EDBC25A2385A8DA9867;Connection: close
```

---

**Fig. 2.** HTTP request preprocessing.
Figure 3 presents the different layers and the hyper-parameters that shape our CNN model. The embedding layer of the CNN transforms $V$ to a digital matrix. The height of the digital matrix $l$ equals to the length of $V$, and its width $d$ defines the size of the embedding vector ($d = 128$). Based on the choice of the kernel size (3, 4, 5, 6), the convolutional layer extracts the features that it considers important. The max-pooling layer selects the features based on an activation function (Relu). These layers reduce the matrix dimension and speed up the computation. Finally, the Softmax layer takes the binary decision and classifies the Http request into two classes. We use 1 to designate the benign requests and 0 to designate the malicious ones.

The neural network architecture presented in Fig. 3 is implemented in order to evaluate our method.

In the next section, we present the evaluation results and compare our technique to the existing works.

![Fig. 3. Implementation of ASCII Embedding technique using CNN.](image)

4 Performance Evaluation of ASCII Embedding

In this section, we evaluate our technique using the two famous splitting methods train/test and k fold cross-validation. We carried out the experiments with the online Http CSIC 2010 dataset [8].
4.1 Dataset Pre-processing

We used the open-access dataset CSIC 2010. It was developed at the “Information Security Institute” of CSIC (Spanish Research National Council) [8]. It was generated to test web attacks protection systems. It contains thousands of Http requests automatically generated: 36000 normal traffic Http requests and 25065 malicious traffic Http requests. This dataset is close to reality as it considers several types of anomalous requests such as SQL injection, buffer overflow, information gathering, files disclosure, CRLF injection, Cross-Site Scripting (XSS), Server-side include, parameter tampering, and Unintentional illegal requests. We split the dataset with the train/test method, we divide the entire dataset into three parts: the training set, the validation set, and the test set. The training and validation set present 80% of the complete CSIC 2010 dataset, and the remaining 20% is for the test. Table 1 shows the distribution of the CSIC 2010 dataset to train, validate, and test the CNN. 63288 requests are used in the training phase and 19413 requests are used to test the efficiency of our method.

| Http requests | Training & validation phase | Testing phase |
|---------------|----------------------------|---------------|
| Normal        | 43236                      | 14400         |
| Abnormal      | 20052                      | 5013          |
| Total number  | 63288                      | 19413         |

The second experiment used the k fold cross-validation method. The goal of this method is to avoid under-fitting and over-fitting problems that can arise with the first method. This experiment aims to validate the results found using the first splitting method. We choose two cases $k = 5$ and $k = 10$. The whole dataset is then divided into $k$ parts. The experiments are repeated $k$ times. In each time, $k - 1$ parts are used for training and one part for testing.

4.2 CNN Hyper-Parameters and Evaluation Metrics

To start the experiments, we choose first the convolutional neural network hyper-parameters. For the depth of the CNN, we choose four different kernel size $(k_1, k_2, k_3, k_4) = (3, 4, 5, 6)$ and $n = 128$ filters per kernel size. We choose Adam as the optimizer, cross-entropy as the loss function, the Relu is the activation function, the batch size is equal to 64, the dropout equal to 0.5, and the embedding vector $d$ equals to 128. To evaluate the effectiveness of our technique, we used accuracy, recall, precision and F1-Score as metrics. These metrics are based on the confusion matrix that determines the value of TP, TN, FP, and FN.
The metrics mentioned above are presented using TP, TN, FP, and FN as follow.

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP}
\]

\[
\text{Sensitivity} = \text{Recall} = \text{True Positive Rate} (TPR) = \frac{TP}{TN + FP}
\]

\[
\text{Specificity} = \text{False Positive Rate} (FPR) = \frac{TN}{FN + FP}
\]

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

\[
F1 - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

Ready to start the experiments, we will present in the next subsection the results.

### 4.3 Experimental Study and Results

Our experiments are conducted using the free cloud Google Collaboratory [15] as platform and software services. It provides an excellent GPU and free deep learning software. Our experiments are conducted using the two different methods of data splitting explained above. The train/test results are presented in the next subsection. The k fold cross-validation results will be presented later.

**Train/Test Split Method.** With 80% of the whole CSIC 2010 dataset, we trained our model CNN using the ASCII embedding method about 6000 steps (about 7 epochs). We recorded the training loss and accuracy rates after every 100 steps. In Fig. 4, we can remark that after 4500 steps of training, the training accuracy rate exceeds 98% while the training losses decrease rapidly towards zero. These trends of accuracy and loss reveal the good performance of our CNN model in the training phase using our new method.

After 6000 steps of training and with 20% of the whole CSIC 2010 dataset, we tested the newly trained CNN using our new method. Table 2 presents the confusion matrix for the binary classification. It presents the different rates TP, TN, FP, and FN. It demonstrates that the ASCII Embedding performs well in the detection of malicious Http requests. Our model CNN based on the ASCII embedding correctly predicts 14282 from 14400 benign requests and 4767 from 5013 malicious requests.
Table 2. Confusion matrix of the binary classification task.

|                  | Predicted as malicious | Predicted as normal |
|------------------|------------------------|--------------------|
| Malicious requests | 4767                   | 246                |
| Benign requests   | 118                    | 14 282             |

Table 3 presents the value of the different metrics obtained from the confusion matrix. Using our model CNN with the new method, the attack detection rate reached: 98.125% of accuracy rate, the precision rate is 99.1805%, the Recall rate is 98.306% and the F1-Score is 96.284%.

Table 3. Performance of ASCII Embedding.

| Metrics | Rate        |
|---------|-------------|
| Accuracy| 98.125%     |
| Recall  | 97.779%     |
| Precision| 94.833%    |
| F1-Score| 96.284%     |

Table 3 reveals that with a convenient amount of training data, our new CNN model based on ASCII Embedding achieved very satisfactory results in detecting web attacks.

The next subsection will present the experiments that used the k fold cross-validation data splitting method.

**k-Fold Cross-validation Method.** Using the Training/Test method for data splitting has some drawbacks. In some cases, either under-fitting or over-fitting can be observed. To avoid these problems, we use the K-fold cross-validation method. This method splits the dataset to k subsets, trains the model with k-1 subsets, and holds the last subset for the test. Consequently, the train/test
phases are repeated k times. In the end, we are sure that each subset is used both in the training and testing phases. In these experiments, we will use as well as the accuracy, the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) [11]. This parameter is a sophisticated criterion for assessing the neural network performance in binary classification problems. Figure 5 presents the performance evaluation of our CNN model using the ASCII embedding method with 5 fold cross-validation. Figure 5(a) shows the different accuracy obtained for k = 5. The higher accuracy achieved 98.49% in round 3 while the less accuracy achieved 97.85% for round 2. The average accuracy obtained is 98.2434%. Figure 5(b) presents the different results of AUC-ROC. It presents the false positive rate (FPR) on the x-axis and the true positive rate (TPR) on the y-axis. It shows that the AUC attains 0.97 which is very closed to 1.

![Fig. 5. Performance evaluation of ASCII embedding using 5-Fold cross-validation.](image)

Figure 6 shows the obtained results using a 10 fold cross-validation method. Figure 6(a) presents the different accuracy found during the experimental tests. The best accuracy equals to 98.44% reached in round 8 while the less accuracy is 97.95% attained in round 7 and 9. The average accuracy reached to 98.2073%. As shown in Fig. 6(b), the Auc reaches 0.97. This value near one proves the considerable performance of the CNN model in detecting web attacks.

![Fig. 6. Performance evaluation of ASCII Embedding using 10-Fold cross-validation.](image)
The experiments we did in this section have two purposes. First, they confirm the efficiency of our technique in preprocessing the input of the model CNN. Second, it outputs the results that will help us compare our technique to the existing ones.

The next subsection will be the scene of the second purpose.

4.4 Comparative Study and Discussion

In the previous subsection, we compared the discrimination ability of our new technique called ASCII Embedding. Table 4 summarizes the results achieved when testing the designed CNN using the new method ASCII Embedding. It shows that the new model CNN performs well in detection web application attacks, the accuracy exceeds 98%.

| Data splitting method          | Accuracy rate |
|-------------------------------|---------------|
| Train/Test                    | 98.125%       |
| 5 Fold cross validation        | 98.2434%      |
| 10 Fold cross validation       | 98.2073%      |

Under the same experimental conditions (dataset, hardware, software) and using the same CNN model (same hyper-parameters values), we implemented and compared the word, and character embedding techniques together with our proposed ASCII embedding. The word and character techniques did not overtake 97.6% and 96.12%, respectively while our approach exceeds 98%. Table 5 presents the obtained results based on the attack detection accuracy criterion. Using the same CNN, we proved by experiments that our CNN based on the ASCII embedding approach achieved better accuracy, it reached 98.2434%.

| Embedding approach | Accuracy  |
|--------------------|-----------|
| Word Embedding     | 97.6%     |
| Character Embedding| 96.12%    |
| ASCII Embedding    | 98.2434%  |

Using the CSIC 2010 dataset, we reveal by all the experiments described above that the ASCII Embedding was successfully used in the web attacks detection. It has a good performance in Http request classification (benign or malicious
It achieved a high web attack detection rate with a low number of false alarms. Our technique uses a simple way to preprocess the neural network input and leads to an improvement of the accuracy rate better than the works using the word or character embedding approaches.

5 Conclusion

In this paper, we pushed a step further the accuracy of neural networks for web attacks detection. We come up with a new method called ASCII Embedding to improve the detection of server-side web application attacks. Using an especially designed convolutional neural network model, our technique outperforms the existing ones and reached 98.2434%. When experimenting with ASCII Embedding, we note an extra waiting time to get the results compared to the character and word embedding techniques.

In future work, we will try to decrease the time overhead. Semantic processing of the Http request can be a possible solution for this problem.

References

1. Prokhorenko, V., Choo, K.-K.R., Ashman, H.: Web application protection techniques: a taxonomy. J. Netw. Comput. Appl. 60, 95–112 (2016)
2. Kumar, K.N., Sukumaran, S.: A survey on network intrusion detection system techniques. Int. J. Adv. Technol. Eng. Explor. 5(47), 385–393 (2018)
3. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education Limited, Malaysia (2016)
4. Xin, Y., et al.: Machine learning and deep learning methods for cybersecurity. IEEE Access 6, 35365–35381 (2018)
5. Lebret, R.P.: Word embeddings for natural language processing. Technical report EPFL (2016)
6. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-aware neural language models. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
7. Ito, M., Iyatomi, H.: Web application firewall using character-level convolutional neural network. In: 2018 IEEE 14th International Colloquium on Signal Processing and Its Applications (CSPA), pp. 103–106. IEEE (2018)
8. Dataset CSIC-2010. http://www.isi.csic.es/dataset/
9. Zhang, M., Xu, B., Bai, S., Lu, S., Lin, Z.: A deep learning method to detect web attacks using a specially designed CNN. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.-S.M. (eds.) ICONIP 2017, Part V. LNCS, vol. 10638, pp. 828–836. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70139-4_84
10. Le, H., Pham, Q., Sahoo, D., Hoi, S.C.: URLNet: Learning a URL representation with deep learning for malicious URL detection. arXiv preprint arXiv:1802.03162 (2018)
11. Yang, S., Berdine, G.: The receiver operating characteristic (ROC) curve. Southwest Respir. Crit. Care Chron. 5(19), 34–36 (2017)
12. Stephan, J.J., Mohammed, S.D., Abbas, M.K.: Neural network approach to web application protection. Int. J. Inf. Educ. Technol. 5(2), 150 (2015)
13. 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)

14. Min, E., Long, J., Liu, Q., Cui, J., Chen, W.: TR-IDS: anomaly-based intrusion detection through text-convolutional neural network and random forest. Secur. Commun. Netw. 2018 (2018)

15. Carneiro, T., Da Nóbrega, R.V.M., Nepomuceno, T., Bian, G., De Albuquerque, V.H.C., Reboucas Filho, P.P.: Performance analysis of Google colaboratory as a tool for accelerating deep learning applications. IEEE Access 6, 61677–61685 (2018)