Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 215 (2022) 687–696

Procedia Computer Science

www.elsevier.com/locate/procedia

4th International Conference on Innovative Data Communication Technology and Application

Web attacks detection using stacked generalization ensemble for LSTMs and word embedding

Rokia Lamrani Alaoui[a], El Habib Nfaoui[b]

[a]PhD student, LISAC Laboratory, Department of Computer Science, Faculty of Sciences Dhar El Mahraz, Sidi Mohamed Ben Abdellah University, Fez, Morocco

[b]Full Professor, LISAC Laboratory, Department of Computer Science, Faculty of Sciences Dhar El Mahraz, Sidi Mohamed Ben Abdellah University, Fez, Morocco

Abstract

Web-based applications are prone to many web security attacks because they are openly-accessible and convenient. Most techniques used to prevent web attacks have some limitations; they cannot detect zero-day attacks and cannot analyze complex attacks, and should be maintained and updated regularly by security experts. Recently, there have been more research work on using deep learning for detecting web intrusions. Moreover, since most high risk web attacks are injected into HTTP web requests, detecting most web attacks needs classifying HTTP web requests into normal and anomalous. In this paper, we propose an approach based on Word2vec embedding and a stacked generalization ensemble model for LSTMs to detect malicious HTTP web requests. We evaluate our classification model performance using the HTTP CSIC 2010 dataset. We show that the combination of different word-level embeddings in a stacked generalization ensemble model for LSTMs has good performance both in terms of classification metrics and training time.

Peer-review under responsibility of the scientific committee of the 4th International Conference on Innovative Data Communication Technologies and Application

Keywords: Web attacks detection; Web security; Deep learning; Word2vec; Stacked generalization ensemble; LSTM

1. Introduction

Nowadays, web applications are omnipresent in our daily lives being used in education, healthcare, and financial institutions. However, they present different drawbacks related to the security and protection of personal data and public goods. Indeed, in a survey conducted in 2019 [24], it was found that 9 of 10 web applications are vulnerable, and that sensitive data breaches are possible on 68% of web applications, and that network intrusions are caused by

* Corresponding author. Tel.: +212-606-570-989 ;
E-mail address: rokia.lamranialaoui@usmba.ac.ma

1877-0509
Peer-review under responsibility of the scientific committee of the 4th International Conference on Innovative Data Communication Technologies and Application
10.1016/j.procs.2022.12.070

unauthorized access to web servers in 8% of cases. Traditional web security techniques, such as static and dynamic code analysis tools and Web Application Firewalls (WAF), cannot detect zero-day attacks and cannot correlate and analyze events for detection of attacks chains. Moreover, WAFs should regularly be updated with the signature of new web vulnerabilities [23].Thus, using web intrusion detection systems based on machine learning and deep learning models [22] is a promising solution to efficient detection of web attacks. In this work, we propose a deep learning based approach for detecting malicious HTTP web requests.
The paper is organized as follows. Section 2 presents a review of related research work. Section 3 presents the necessary background. Section 4 explains the proposed approach. Section 5 presents and discusses the experimental results. Conclusion and future work are shown in the last section.

2. Literature Review

In this section, we review some existing literature in web vulnerabilities detection based on deep learning. [19] use word-level embedding, manually extracted statistical features, along with CNN and GRU networks to detect web anomalies; the model achieves 97.56% and 99.00% accuracy over the HTTP CSIC 2010 dataset, without and with features augmentation respectively. [14] detect 95% of web attacks in the HTTP CSIC 2010 dataset by combining CNN, LSTM networks, and one-hot encoding. [21] use CNN networks and character-level embedding for detecting malicious URLs, file paths and registry keys. They obtain 92% accuracy at 0.1% false positives on the detection of malicious URLs in a custom dataset. [11] achieve 99% accuracy over the CW900 public dataset by using ResNN and GRU networks to detect websites fingerprinting attack. [16] propose a CNN based model to prevent SQL injection attacks. The model achieves 99.50% accuracy on a custom dataset. [20] use a character-level embedding and CNN networks to detect SQLI, Directory Traversal, Remote File Inclusion and XSS. The model achieves 0.02% false positive on a custom dataset. [2] achieve 99.97% accuracy on the HTTP CSIC 2010 dataset by using LSTM networks to develop intrusion detection systems. [18] use 2-grams method to pre-process HTTP requests, DBN and stacked auto-encoder to extract most relevant features, and the 1-class SVM algorithm to classify HTTP web requests. They show that deep neural networks used as features extractors can help improving traditional machine learning models performance. [13] use CNN, LSTM and DNN, along with UTF-8 encoding to detect malicious URLs in online realtime data. In [15], the authors propose URLNet; a framework for detecting malicious URLs by applying CNN to char-level word embedding which is a novel approach to text vectorization. They experimentally show that URLNet outperforms Word-level CNN and character-level CNN. In [17], the authors fed textual and statistical features into an ensemble classification model composed of LSTM, SVM, Random Forest and Logistic regression. They show that the ensemble model performs better than contributing models in most cases. Finally, in [1], the authors performed a systematic Literature review on Deep Learning based web attacks detection, which compiles research papers published between 2010 and 2021.

3. Preliminaries

3.1. Problem definition

Web applications are prone to many web attacks. Yet, each web attack has its own characteristics that it is difficult to extract features that capture the particularities of all attacks. However, HTTP web requests is the most channel attack used by web intruders to launch the most critical web attacks. Thus, detecting web attacks needs classifying HTTP web requests into malicious and benign. In this work, we formulate the problem of web attacks detection as a binary classification of HTTP web requests. A malicious HTTP web request indicates the presence of a web attack while a benign HTTP web request indicates its absence. We make use of word embedding to transform HTTP web requests to vectors and then train a stacked generalization ensemble for deep learning models to distinguish between normal and anomalous HTTP web requests.

3.2. Deep learning techniques

Machine learning aims at constructing algorithms that can learn from data and make predictions on data. Deep learning methods are a subfield of machine learning. In the following, we highlight different deep learning models that we will use in our experiments. **Convolutional Neural Networks (CNN)**: CNN have been commonly used in image recognition and NLP problems. The advantage of CNNs over fully connected neural networks is that they can learn less parameters for the same performance which results in faster training and will help to build deep CNNs [4]. Also, CNNs can extract complex and relevant features by applying convolution and pooling operations on raw data. **Recurrent Neural Networks (RNN):** In RNNs, the behavior of the network varies over time: hidden neurons may depend not only on neurons in previous layers but also in neurons at earlier times. This property makes of RNNs a good solution for processing sequential information. There are a lot of RNN variants, the most common used are LSTM [12] and GRU [6] which were developed to address the vanishing gradient problem. **Stacking ensemble for deep neural networks:** In stacked generalization ensemble [26] for deep learning neural networks, a deep learning model called meta-model is trained how to best combine the predictions of different deep learning sub-models to make a better output prediction. In order to build a stacked ensemble model, the dataset is split into a train, validation and test sets. Then, the sub-models are fit on the train set, the meta model is fit on the test set, and the evaluation of the stacking ensemble model is done on the validation set. Stacked generalization, or stacking for short, can result in better predictive performance than any single contributing sub-model.

3.3. Word embeddings

There are many word representations for natural language understanding tasks; Bag Of Words, TF-IDF, GloVe and others. In this paper, we will use Word2vec embedding to generate vector representations of HTTP Web requests.

4. Proposed approach

4.1. Overview

Figure 1 shows the steps followed by the proposed web attacks detection method. Firstly, we select the features that are relevant for the classification task. Secondly, we concatenate the selected features in order to have a textual format of the input. Thirdly, we pre-process the resulting input (see section 4.2). Then, we use different word2vec models to get different word embeddings of the pre-processed HTTP Web request. We pass the word embeddings to a stacked generalization ensemble for LSTM models which then classifies HTTP web requests.

4.2. Preprocessing

4.2.1. Feature selection and concatenation:

HTTP web requests consists of different elements. We select the HTTP method which is a verb that defines the operation the client wants to perform, the request or the URL which is the path of the resource to fetch, and the payload which contains the data submitted by the client. We concatenate the three elements so to have a textual input.

4.2.2. Decoding:

Overall, web attackers may be able to bypass traditional web application firewalls by using different encoding techniques such as URL encoding, base64 encoding, Hex encoding etc. For this reason, we URL decode the input data so to obtain its original form.

4.2.3. Generalization:

We generalize the decoded data as follows: Firstly, we replace the various “http://name-of-website” with “http://u” or “https://u”. Then, all the numbers in the input data will be replaced with “0”. This step improves the quality of the input data by reducing meaningless information.

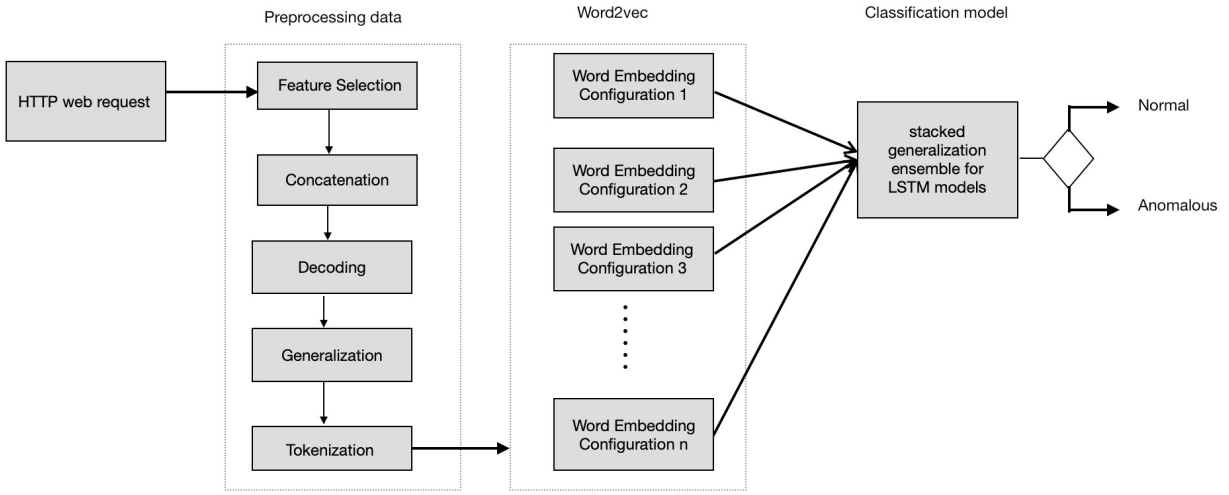

Fig. 1. Flow graph of proposed approach

4.2.4. Tokenization:

We use custom regular expressions to split the input data to a set of tokens that we use as a text corpus to train word2vec models.

4.2.5. Vectorization: Word2vec

Word2vec is a two-layer neural network that takes a text corpus as input and outputs a set of vectors. It includes two models; CBOW and Skip-Gram. We used CBOW model because it is usually faster and more accurate than the Skip-Gram model. We modified training parameters values, namely the window size and embedding size, to obtain different vectors representations of the same word.

4.3. Classification: Stacked generalization ensemble of LSTM models

We propose a stacked generalization ensemble for LSTMs to detect malicious HTTP web requests. Figure 2 describes the proposed classification process: we pass the HTTP web request through different word2vec models in order to obtain different word embeddings. Then, each sub-model, which is an LSTM network, takes one of the resulting word embeddings and outputs a prediction of whether the HTTP request is normal or malicious. Afterwards, the meta model, which is a shallow neural network, combines the predictions from each input sub-model and returns the final prediction.

5. EXPERIMENTS AND RESULTS

We implemented and compared models based on LSTM, CNN and stacking ensemble in order to choose the most accurate model. We mention that each experiment is run 5 times and the average value of each performance metric is reported. The code of the experiments is published in our GitHub repository [10].

5.1. Dataset

We use the HTTP CSIC 2010 dataset which contains 223,585 samples. We randomly split the dataset into three groups such as there is no class imbalance in each group. The training set, the validation set and the test set represents 67%, 16% and 17% of the dataset respectively.

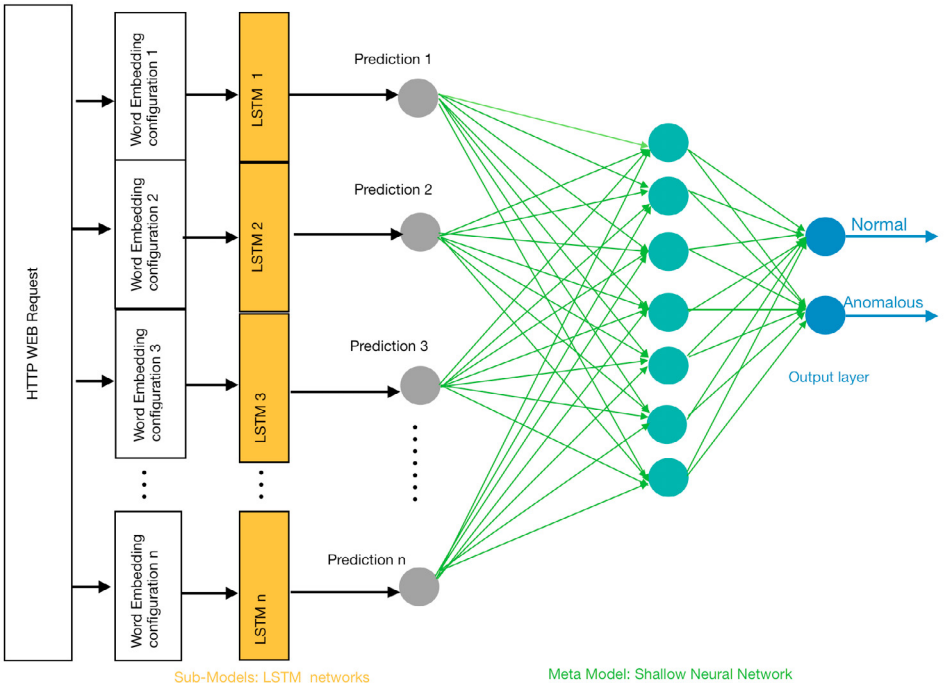

Fig. 2. Proposed model architecture

5.2. Data pre-processing

5.2.1. Feature Selection and concatenation:

The HTTP CSIC 2010 dataset contains features like protocol, acceptEncoding, acceptLanguage, user-agent, etc which are redundant and not relevant. Thus, we only keep three features which are the HTTP method, the HTTP request and the payload. Next, we concatenate the values of the three features so to obtain the format shown in Table 1. In this way, we can exploit vectorization techniques used in NLP problems(e.g. [8]) to convert HTTP web requests to numerical vectors.

Table 1. Example of a record after feature selection and concatenation

HTTP request	Class
GET http://localhost:8080/vtibin/fpcount.exe?	anom

5.2.2. URL Decoding:

We apply URL decoding in order to obtain a de-obfuscated string. For instance, an HTTP web request that contains "*B2 = bob%2540%253CSCRipt%253Ealert%2528Paros%2529%253C%252FscrIPT%253E.parosproxy.org*" is restored to "b2=bob@<script>alert(paros)</script>. parosproxy.org"

5.2.3. Generalization and Tokenization:

We replace words that have different syntax but share the same meaning with a unique word. Afterwards, we tokenize the generalized HTTP web request by using a custom function which properly splits the web request into an array of words.

5.2.4. Word2vec based vectorization:

It consists in converting each word in the array returned by the tokenizer to a vector of d numerical values; d is called the embedding dimension. Thus, if a web request contains s words, the dimension of the corresponding embedding matrix is $s \times d$. We distinguish between static and non-static word embedding. In static word embedding, vector representations of words do not change during the training process while in non-static word embedding they are modified in order to improve the classification model performance. In our experiments we use static word embedding to avoid overfitting the training data. Also, in the context of web attacks, there do not exist pre-trained Word2vec models. Hence, we extract a corpus of words from the HTTP CSIC 2010 dataset and train Word2vec models to learn word associations from the entire corpus; however, in order to approximate real life applications, we create an index for words that are part of the training set only.

5.3. Evaluation metrics:

We use four performance metrics; accuracy, precision, recall and F1-score, which are common in the evaluation of deep learning based intrusion detection systems.

5.4. Deep learning models architecture

We develop four deep learning models. Two models use CNN and LSTM networks and the two others use stacking ensemble of LSTM or CNN networks. All the models take as input different word-level embeddings obtained from various Word2vec models that we built using different values of word2vec training parameters. The four models differ in that the two first models operate on a single word-level embedding at a time, while the two others operate on multiple word-level embeddings simultaneously. Since the dataset is not too large, we take care to not use very complex models. Also, we use two regularization techniques, namely layer normalization and dropout, in order to improve the performance of the models. More detail on the DL models structure is given in the corresponding Experimental results section 5.6.

5.5. Experimental environment

All the models are implemented using Keras, TensorFlow, gensim, scikit-learn, numpy and pandas and they are run on Google Collaboratory platform. In tables 2 and 3, we give a summary of parameters which values remain the same through all the different experiments.

Table 2. Parameters settings for training deep learning models

Parameter	Value
activation function in hidden layers	tanh (LSTM) and relu (CNN)
activation function in output layers	softmax
loss function	categorical cross entropy
number of epochs	20
batch size	128
optimizer	Adam
learning rate	0.001

5.6. Results

5.6.1. A. Experiment 1.

We implemented a 3-layers CNN and we applied different window sizes to train the word2vec model. We used the embedding dimension equal to the number of feature maps. Table 4 describes the results of the different experiments (the results in bold correspond to the best experiment). The results show that while using the same kernel sizes, and equal number of feature maps and embedding dimensions, the performance metrics values improve significantly with increasing window sizes.

Table 3. Parameters settings for training Word2vec models

Parameter	Value
number of epochs	30
learning algorithm	CBOW
learning rate	0.1
Maximum URL length	20

Table 4. Experimental results of Model 1- 3-layers CNN and one fully connected layer

Experiment	kernel size	Feature maps	Embedding size	window size	Accuracy%	Precision%	Recall%	F1-score%
1	2, 3, 4	128,128,128	128	2	77.89			
2	2, 3, 4	128,128,128	128	3	77.98			
3	2, 3, 4	128,128,128	128	4	78.004			
4	**2, 3, 4**	**128,128,128**	**128**	**5**	**78.009**	**81.4**	**77.17**	**75.6**

5.6.2. B. Experiment 2.

We implemented a CNN stacking ensemble model and we changed the embedding dimension or the window size provided that they are equal to the number of feature maps and the kernel size respectively. Table 5 describes the results of the different experiments (the results in bold correspond to the best experiment). We remark that there is no significant difference in classification results when different window sizes or embedding dimensions are used. However, the training time increases significantly with larger embedding dimensions and feature maps.

Table 5. Experimental results of Model 2 - stacking ensemble of CNN models

Experiment	kernel size	Feature maps	Embedding size	window size	Accuracy%	Precision%	Recall%	F1-score%
1	2	300	128	5	78.20			
	3	300	128	5				
	4	300	128	5				
	5	300	128	5				
2	2	300	128	2	78.12			
	3	300	128	2				
	4	300	128	2				
	5	300	128	2				
3	2	256	256	2	78.17			
	3	256	256	3				
	4	256	256	4				
	5	256	256	5				
4	**2**	**128**	**128**	**2**	**78.24**	**81.5**	**78.4**	**77.5**
	3	**128**	**128**	**3**				
	4	**128**	**128**	**4**				
	5	**128**	**128**	**5**				

5.6.3. C. Experiment 3.

We implemented a 3-Layers LSTM model and we used four different window sizes; 2, 3, 4 and 5, for each of three different embedding dimensions; 128, 256 and 512. Table 6 summarizes the results of the different experiments. We can see from the results that for the same number of LSTM units, the classification results of the models improve when we use the window size that suits the best the chosen embedding dimension. But, overall, the results get better with increasing window sizes.

5.6.4. D. Experiment 4.

We implemented an LSTM stacking ensemble model and we changed either the embedding dimension or the window size or both. Table 7 describes the experimental results. We observe that using different window sizes and

Table 6. Experimental results of Model 3- 3-layers LSTM and one fully connected layer

Experiment	Number of units	Embedding size	window size	Accuracy%	Precision%	Recall%	F1-score%
1	128, 256, 512	128	2	77.893			
2	128, 256, 512	128	3	77.890			
3	128, 256, 512	128	4	78.00			
4	**128, 256, 512**	**128**	**5**	**78,21**			
5	128, 256, 512	256	2	78.08			
6	128, 256, 512	256	3	78.18			
7	**128, 256, 512**	**256**	**4**	**78.25**	**80.94**	**77.56**	**77.65**
8	128, 256, 512	256	5	78.20			
9	128, 256, 512	512	2	78.12			
10	128, 256, 512	512	3	78.14			
11	128, 256, 512	512	4	78.07			
12	**128, 256, 512**	**512**	**5**	**78.16**			

embedding dimensions is better than using the same window size or embedding dimension. Also, even if different window sizes are used it is important to find the good combination between embedding dimension, window size and number of LSTM units.

5.7. Discussion

In our experiments, we implemented LSTM and CNN models and compared their performance when used as individual classifiers and as part of a stacked generalization ensemble model. We generated various words embeddings by training word2vec models with different values of word2vec training parameters, namely the window size and the embedding dimension. In terms of execution time, training stacking ensemble models is, overall, faster than training LSTM or CNN models separately. Also, thanks to GPU implementation of LSTM networks, training LSTM models is faster than training CNN models. As for the classification performance, we can see that stacking ensemble for LSTM models performs better with 78,95% accuracy,81.54% precision, 78.41% recall, 77.57% F1-score against 78,25% accuracy, 80.94% precision, 77.56% recall, 77.65% F1-score for LSTM model. As for CNN based models, the stacking ensemble for CNN models is also better than CNN model with 78,24% accuracy , 81.5% precision, 78.4% recall, 77.5% F1-score against 78,00% accuracy,81,4% precision, 77.17% recall, 75.6% F1-score. This can be explained by the fact that since the performance of word2vec models is evaluated based on the underlying classification task, combining different Word2vec models in a stacked generalization ensemble model strengths the classification results as different vector representations are involved in the training process. We also observe that LSTM model and stacking ensemble for LSTM models outperform both CNN model and stacking ensemble for CNN models. Moreover, the experimental results show that in order to improve the classification models performance, it is important to find out the best combination between the embedding dimension, the window size and the number of LSTM units or the kernel size and the number of feature maps in CNN layers.

6. CONCLUSION

In this paper, we developed a stacked generalization ensemble for LSTMs to detect malicious HTTP requests. The embedding layer of each sub-model is learned from a word2vec model that was trained using different configurations of training parameters. The experimental results show that this model has the best classification results and the least training time compared with CNN, LSTM networks and stacking ensemble for CNN models. In future work, we will study how to leverage other sentence classification techniques to enhance the detection of web attacks. Also, it would be interesting to evaluate the performance of the proposed model in other information security contexts, namely malware detection (e.g. [7], [25]), Network intrusion detection (e.g. [5], [9]), and IoT Networks (e.g. [3]).

Table 7. Experimental results of Model 4 - Stacking ensemble of LSTM models

Experiment	Number of units	Embedding size	window size	Accuracy%	Precision%	Recall%	F1-score%
1	256	512	5	78.50			
	128	256	5				
	64	128	5				
	32	64	5				
2	256	512	4	78.70			
	128	256	4				
	64	128	4				
	32	64	4				
3	256	512	3	78.58			
	128	256	3				
	64	128	3				
	32	64	3				
4	256	512	2	78.64			
	128	256	2				
	64	128	2				
	32	64	2				
5	256	512	4	78.37			
	128	256	3				
	64	128	5				
	32	64	2				
6	**256**	**512**	**5**	**78.95**	**81.54**	**78.41**	**77.57**
	128	**256**	**4**				
	64	**128**	**3**				
	32	**64**	**2**				
7	256	128	5	78.05			
	128	128	4				
	64	128	3				
	32	128	2				
8	512	512	5	78.67			
	256	256	4				
	128	128	3				
	64	64	2				
9	128	256	5	78.49			
	64	128	4				
	32	64	3				
	16	32	2				

References

[1] Alaoui, R.L., Nfaoui, E.H., 2022. Deep learning for vulnerability and attack detection on web applications: A systematic literature review. Future Internet 14, 118.

[2] Althubiti, S., Nick, W., Mason, J., Yuan, X., Esterline, A., 2018. Applying long short-term memory recurrent neural network for intrusion detection.

[3] Amanullah, M.A., Habeeb, R.A.A., Nasaruddin, F.H., Gani, A., Ahmed, E., Nainar, A.S.M., Akim, N.M., Imran, M., 2020. Deep learning and big data technologies for iot security. Computer Communications 151, 495–517.

[4] A.Nielsen, M., 2019. Neural Networks and Deep Learning.

[5] Ashiku, L., Dagli, C., 2021. Network intrusion detection system using deep learning. Procedia Computer Science 185, 239–247. URL: https://www.sciencedirect.com/science/article/pii/S1877050921011078, doi:https://doi.org/10.1016/j.procs.2021.05.025. big Data, IoT, and AI for a Smarter Future.

[6] Chung, J., Gulcehre, C., Cho, K., Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 .

[7] Duraipandian, M., Vinothkanna, M.R., 2019. Machine learning based automatic permission granting and malware identification. Journal of Information Technology 1, 96–107.

[8] Elfaik, H., et al., 2019. A comparative evaluation of classification algorithms for sentiment analysis using word embeddings, in: International Conference on Advanced Intelligent Systems for Sustainable Development, Springer. pp. 1–11.

[9] Fu, Y., Du, Y., Cao, Z., Li, Q., Xiang, W., 2022. A deep learning model for network intrusion detection with imbalanced data. Electronics 11, 898.
[10] GitHub, 2020. Implementations code. https://bit.ly/2HNXPzw.
[11] He, X., Wang, J., He, Y., Shi, Y., 2018. A deep learning approach for website fingerprinting attack, in: 2018 IEEE 4th International Conference on Computer and Communications (ICCC), pp. 1419–1423.
[12] Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.
[13] Kim, A., Park, M., Lee, D.H., 2020. Ai-ids: Application of deep learning to real-time web intrusion detection. IEEE Access 8, 70245–70261.
[14] Kuang, X., Zhang, M., Li, H., Zhao, G., Cao, H., Wu, Z., Wang, X., 2019. Deepwaf: Detecting web attacks based on cnn and lstm models, in: International Symposium on Cyberspace Safety and Security, Springer. pp. 121–136.
[15] Le, H., Pham, Q., Sahoo, D., Hoi, S.C., 2018. Urlnet: Learning a url representation with deep learning for malicious url detection. arXiv preprint arXiv:1802.03162 .
[16] Luo, A., Huang, W., Fan, W., 2019. A cnn-based approach to the detection of sql injection attacks, in: 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), pp. 320–324.
[17] Mišík, D., Hudec, L., 2019. Detection of intrusions to web system using computational intelligence, in: International Conference on Information Systems Architecture and Technology, Springer. pp. 199–208.
[18] Moradi Vartouni, A., Teshnehlab, M., Sedighian Kashi, S., 2019. Leveraging deep neural networks for anomaly-based web application firewall. IET Information Security 13, 352–361.
[19] Niu, Q., Li, X., 2020. A high-performance web attack detection method based on cnn-gru model, in: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), pp. 804–808.
[20] Rong, W., Zhang, B., Lv, X., 2019. Malicious web request detection using character-level cnn. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) , 6–16.
[21] Saxe, J., Berlin, K., 2017. expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys. arXiv preprint arXiv:1702.08568 .
[22] Sommer, R., Paxson, V., 2010. Outside the closed world: On using machine learning for network intrusion detection, in: 2010 IEEE symposium on security and privacy, IEEE. pp. 305–316.
[23] Sonewar, P., Thosar, S., 2017. Detection of sql injection and xss attacks in three tier web applications.
[24] Technologies, P., 2020. Web applications vulnerabilities and threats: statistics for 2019. https://www.ptsecurity.com/ww-en/analytics/web-vulnerabilities-2020/.
[25] Vivekanandam, B., 2021. Design an adaptive hybrid approach for genetic algorithm to detect effective malware detection in android division. Journal of ubiquitous computing and communication technologies 3, 135–149.
[26] Wolpert, D.H., 1992. Stacked generalization. Neural networks 5, 241–259.