Deep Learning within the Web Application Security Scope – Literature Review

Matija Kaniški, Jasminka Dobša and Dragutin Kermek
Faculty of Ogranization and Informatics, University of Zagreb, Varaždin, Croatia
{matija,kaniski, jasminka.dobsa, dragutin.kermek}@foi.unizg.hr

***Abstract* - Over the last few years, several breakthroughs in deep learning have contributed to the development of new models. One of many areas they are applied to is the web application security scope. Web applications are still one of the biggest information and business security threats. Requests sent to the Web application are divided into normal and malicious. Malicious requests contain a payload that exploits a discovered vulnerability. Detection of Web attacks can be reduced to natural language processing classification problem. Lately, pre-trained models on Transformer neural networks showed promising results in the detection of Web attacks. In development of models, the preprocessing step of data preparation is crucial. After preparation of good datasets and application of powerful models it is very important to evaluate and compare performance of algorithms. The goal of this paper is to conduct an overview of the deep learning methods used for Web attack detection. The research is conducted by querying scientific databases, analyzing relevant articles within the security scope, and summarizing the proposed state-of-the-art approaches. Findings of reviewed papers were summarized based on implementation details and used performance metrics. Also, open problems will be emphasized, as well as challenges and possibly new opportunities for the future research.**

Keywords - deep learning; transformer architecture; natural language processing; web application security; web attack detection

I. INTRODUCTION

Web applications are software systems that generate web pages and documents written in one of the programming languages and are executed on the server. Web browsers are programs through which we access the Web application. Requests sent to the Web application are divided into normal and malicious payloads. Malicious requests contain a payload that exploits a discovered vulnerability. With the emergence of new Web technologies, modules, frameworks and platforms, Web applications open new attack surfaces and vectors. Attack surface is what is being attacked, whereas an attack vector is a path or means by which an attacker can gain access to a computer system or network to deliver a malicious payload. Such a payload is sent to the Web application, and the response is analyzed for possible vulnerabilities. Vulnerability is a known or unknown weakness (zero-day) that can be exploited. Further, exploits take advantage of that vulnerability to gain unauthorized access or execute malicious tasks. In the last decade neural networks have regained popularity [1] due to the rapid development of hardware.

Also, large amounts of data provide new opportunities in their use, including natural language processing (NLP). Deep learning (DL) is a part of machine learning (ML) where neural networks are essential components of the algorithm [2]. Determining an attack on a Web application with DL methods is a classification task of whether the payload sent to the Web application is malicious or not. Therefore, the detection of Web attacks can be reduced to an NLP classification problem. In development of models preprocessing step of data preparation is crucial. After preparation of good datasets and application of powerful models it is very important to evaluate and compare performance of algorithms. The motivation for the research originates from practical applications of training neural networks to improve Web attack detection. Existing literature reviews [1, 2] cover DL models for a specific Web attack like denial of service (DoS) or analyze multiple ML and DL techniques for the detection of unknown (known as zero-day) Web attacks. This study considers DL models for Web attack detection that are not limited to specific types of Web attacks or devices (IoT). In addition, payloads divided as normal or malicious (binary classification) will be observed. This paper is structured in the following way. Section I is the introduction. Section II focuses on defining research questions, strategy, and criteria. Section III brings up the analysis of selected papers. Section IV contains research findings and a discussion. Section V concludes the paper.

II. METHODOLOGY

This paper analyzes and summarizes DL methods for Web attack detection within the Web application security scope. It is crucial to identify primary studies and conduct an overview of different approaches for classifying malicious payloads to achieve this objective.

A. Defined research questions

Insufficiently described or omitted factors in the DL literature lead to difficulty reproducing or replicating a given approach. Major contributing factors to these issues are the lack of published open-source implementations and datasets, missing data filtering details and descriptions of hyperparameters used [1,2]. The research questions are: (i) What types of Web attacks have been addressed by DL-based approaches? (ii) How are artifacts being extracted, prepared, and used in DL-based approaches for Web attack detection? (iii) What DL models are used to support Web attack detection? (iv) How well does DL-based approach perform in supporting various Web attack detection?

B. *Search strategy*

The search was conducted on three scientific databases: Web of Science, Scopus, and IEEE Xplore. Title, abstract and keywords in the Computer Science research topic were searched. The search query was adjusted several times to find only those papers that are within the scope of the research. Defined search keywords were *deep learning*, *neural network*, *web attacks*, *web security*, *web application security* and *web vulnerabilities*.

Each database had different search queries according to their search engine. The following search queries yielded the most relevant results:

- **Web of Science**: TOPIC: (“deep learning” OR “neural networks”) AND (“web attacks” OR “web security” OR “web application security” OR “web vulnerabilities”)
- **Scopus**: TITLE-ABS-KEY ((“deep learning” OR “neural networks”) AND (“web attacks” OR “web security” OR “web application security” OR “web vulnerabilities”)) AND (EXCLUDE (PUBYEAR, 2023)) AND (LIMIT-TO (SUBJAREA, “COMP”)) AND (LIMIT-TO (LANGUAGE, “English”))
- **IEEE Xplore**: ALL: (“deep learning” OR “neural networks”) AND (“web attacks” OR “web security” OR “web application security” OR “web vulnerabilities”)

All queries aim to extract relevant articles about Web attack detection with DL-based approaches.

C. *Search criteria*

Selection criteria are presented in Table I. These inclusion criteria are used to determine which paper is relevant and which is not. The table also clearly states all exclusion criteria for further reducing the papers.

TABLE I. INCLUSION AND EXCLUSION CRITERIA

Inclusion criteria	***Exclusion criteria***
Papers written	Non-English papers
Papers published between 2016 and 2022	Papers published before 2016 and after 2022
Papers published in scientific conferences or scientific journals or relevant Web articles	Duplicate papers
Papers on DL for Web attack detection	Out of scope papers
Papers which proposed new models or approaches	Papers who do not provide sufficient details about their models or approaches

III. OVERVIEW OF INCLUDED PAPERS

Applying the queries leads to filtering all papers considering the constraints. The section will show the number of articles per database, per year, and by paper type. Afterward, the analysis is presented. In this section is presented the number of papers selected in each iteration. There were four iterations whose results are displayed in Figure 1.

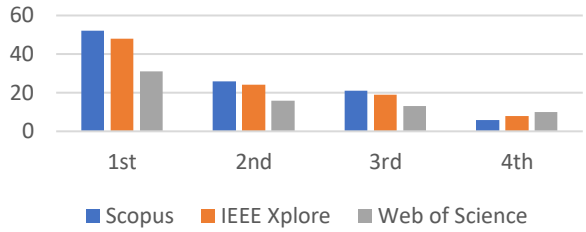

Figure 1. Number of papers per iteration and scientific database

In the first iteration queries are executed to get relevant articles without duplicates. In the second iteration it is done filtering based on reading titles and abstracts of the papers. The third iteration it is done filtering after reading full articles. The last iteration is based on narrowing the scope and focusing on answering all the research questions. After the process of selection, a total of 24 papers were analyzed and considered. Figure 2 shows the number of papers that used deep learning for detecting Web attacks in the observed period by years.

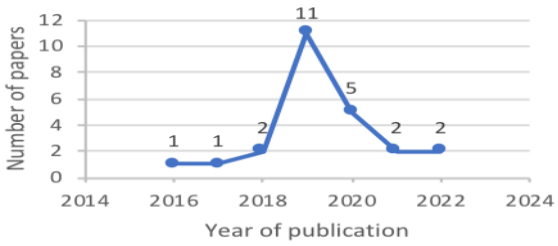

Figure 2. Number of selected papers per publication year

As one can see, most articles were published in 2019, and then the interest decreased. The reason for this decrease is probably the appearance of transformer models with self-attention mechanisms that yielded state-of-the-art performance in Web attack detection. The distribution of published research papers per year suggests that topics regarding Web attack detection with deep learning are an active area for research. We predict that it will still be a desirable area in the future since new and unknown Web attack variants are emerging which could be detected by new approaches using neural language models. Paper classification based on its type is shown in Figure 3.

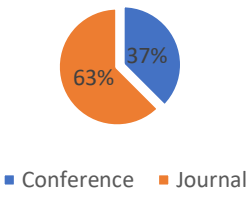

Figure 3. Precentage of papers per type of publication

Not surprisingly, the percentage of selected journal articles is higher than that of conference papers. Mainly because to answer all research questions, additional details of the proposed approach must be provided, which was not the case in most conference papers.

IV. RESULTS AND DISCUSSION

The existence of vulnerabilities doesn't mean it will be exploited. Only after a few years of initial discovery a particular attack technique is becoming widely misused. That's why we are exploring them right now. Table II summarizes the reviewed papers in six categories. Approaches were mainly supervised learning models (SLM) and unsupervised learning models (USLM).

A. Type of Web attacks

Injection Web attacks are considered to impact the most vulnerabilities in Web applications. Injection of HTTP payload [3-8] show different patterns hidden in the URL. Next, there are SQL injection (SQLi) and Cross-site scripting (XSS) [9-19]. SQLi attacks are server-side vulnerabilities targeting Web application databases, whereas XSS are client-side vulnerabilities targeting Web application users. Then there is XPath [20], a query language for XML with injection issues similar to SQL. Another Web attack is phishing [21-22]. Phishing is the malicious practice of luring users into disclosing their personal information. Fake websites may be created and employed by criminals to steal that information. Further, there are denial of service (DoS) attacks whose purpose is to make a system unavailable for access. In contrast to DoS attacks, distributed denial of service (DDoS) attacks [23-24] use several computers to execute a simultaneous DoS attack against one or more systems. Webshells (backdoors) [25] of the website are often implemented on the website servers to maintain the management authority. Web trojan [26] attacks are designed to damage, disrupt, and inflict harmful actions, like personal information leakage. Zero-day Web attacks are unknown attacks that have not been reported. They are very rare and hard to find in the HTTP requests. Web application firewalls (WAF) are a common defense against Web attacks, but they can be exploited to evade the filter detection pattern by leveraging the found vulnerability.

B. Datasets

The lack of public datasets in the Web application security scope represents an obstacle to training and testing DL algorithms. Most often publicly available datasets in the reviewed papers are:

- CISC2010 dataset [3-7, 10, 11, 15, 24] has automatically generated traffic with multiple Web attacks like buffer overflow, information gathering, SQLi, XSS, files disclosure, CRLF injection, parameter tempering and more.
- ECMLPKD dataset [5] contains more than 35000 normal requests and about 15000 malicious Web requests collected from real network traffic.
- httpParams dataset [7, 15] is created with several tools like sqlmap, xssya, Vega Scanner, FuzzDB.
- FWAF dataset [7, 15] is created by FSECURITY for detecting malicious Web queries with ML-driven WAF.
- XSSed dataset [16] contains more than 40000 different malicious XSS attacks.
- UCI phishing dataset [21, 22] consists of about 11000 samples which have 30 features pre-classified as phishing or normal.
- CICIDS2017 dataset [24] contains DDoS attacks and 83 features in the dataset, including both the network traffic and package information results.
- KDDCU99 dataset [24] contains samples with four types of anomalies DDoS, Remote-to-login, User-to-root, and probing.

Private data sets are also used. A dataset is considered private if it is not publicly available or if no procedure is described to replicate and reproduce it. Private datasets [3-5, 8-14, 17-20, 23, 25, 26] contain new examples, normal or malicious requests, on which the model was trained.

C. Preprocessing techniques

Original datasets must be transformed into numerical data by procedure called feature extraction. Not all features are equally important for the classification decision. Feature selection is an essential pre-processing task that aims to find a subset of relevant features from the original features. In examined papers, different data preprocessing techniques have been used, where tokenization and neural embedding are by far the most common. Also, encoding techniques such as HEX, URL, Unicode, HTML entity and generalization are used to reduce interference of redundant and irrelevant information. Tokenization [4, 11, 13, 15-17] is a broad technique that was applied to separate sequential data into tokens depending on the type of Web attack. Neural embeddings [8, 9, 22, 24, 26] use different DL architectures to represent the data. Two popular implementations of Wor2Vec [12, 14, 16] used for feature extraction are CBOW model [7, 10, 18, 25] and the Skip-Gram model [6]. Some papers did not explicitly mention the type of embedding they use: word-level [4, 5] or character-level embedding [3, 20]. Besides that, feature selection and weighting using Generic Algorithm (GA) [21] and statistical feature extraction [23] were present in the pre-processing task.

D. Classification model

Each model has its strengths and weakness in detecting certain types of Web attacks. CNN model [3, 8, 10, 16, 25] and its variants [6, 8, 11, 14, 24] have the most significant advantage in that its feature set is learned by itself. RNN [16, 20] is a DL model suitable for solving the sequential problem, identifying patterns in URLs. LSTM [3-5, 10, 20] is a special RNN to realize the long-term memory of characters. Compared with RNN, CNN training is shorter, whereas CNN compared with a deep neural network (DNN), its parameters are fewer, and the model is more concise. MLP [3, 10, 13, 15, 17] is a feed-forward neural network (FFNN) with multiple processing layers and at least one hidden layer, among which both the hidden layer and the output layer have the ability for Web attack detection. Autoencoder is an FFNN with one or more layers designed to minimize input and output differences. A stacked Autoencoder (SAE) [9, 26] is constructed by stacking the input and hidden layers of the Autoencoder layer by layer.

TABLE II. SUMMARY OF REVIEWED PAPERS

Ref	Implementation details of the proposed approach							Performance by most frequent performance metrics			
	Web attack	***Type***	***Framework***	***Dataset***	***Prep. technique***	***Over/under -fitting protection***	***Model***	***Acc. (%)***	***TPF (%)***	***F1-score (%)***	***FPR (%)***
[3]	Multiple Web attacks injected in HTTP request or payload	SLM	PyTorch	**CSIC2010***, Private	Character embedding	Data cleaning	MLP, CNN, LSTM	97.79	96.04	98.72	N/A
[4]		SLM	N/A	**CSIC2010***, Private	Tokenization, Word embedding	Data balancing	LSTM-MLP	98.42	97.56	N/A	N/A
[5]		SLM	TensorFlow / Keras	**CSIC2010***, ECMLPKD, Private	Word embedding	Data augmentation	Bi-LSTM	98.6	98.4	98.1	N/A
[6]		SLM	N/A	CSIC2010	Skip-gram	Boosting	Text CNN SVM	99.33	99.09	N/A	N/A
[7]		SLM	TensorFlow / Keras	CSIC2010, httpParams, FWAF	CBOW**,** FastText	Dropout	ResNet	99.41	99.55	N/A	N/A
[8]		SLM	TensorFlow / Keras	Private	Neural embedding	Early Stopping	CNN, GRU	99.61	99.58	99.61	N/A
[9]	SQLi, XSS, object deserial.	USLM	TensorFlow / Keras, WEKA	Private	Neural embedding	Denoising	SAE	N/A	92.8	91.8	N/A
[10]	SQLI, XSS	SLM	N/A	**CSIC2010***, Private	CBOW	Dropout, Data cleaning	Ensemble MLP, CNN, LSTM	99.47	99.70	99.0	0.33
[11]		SLM	N/A	**CSIC2010***, Private	Tokenization	Boosting, Data balancing	CNN with attention	99.845	98.29	99.1	0.0
[12]	SQLi	USLM	N/A	Private	Word2Vec	Data cleaning, Data balancing	Encoder-Decoder RNN	N/A	99.0	98.0	N/A
[13]		SLM	PyTorch	Private	Tokenization	Boosting	MLP	99.75	99.88	N/A	0.02
[14]		SLM	TensorFlow / Keras	Private	Word2Vec	Data cleaning	Elastic Pooling CNN	99.93	99.94	99.95	N/A
[15]		SLM	PyTorch	CSIC2010, httpParams, FWAF	BERT tokenizer	Data balancing	MLP	99.98	N/A	98.7	N/A
[16]	XSS	SLM	TensorFlow / Keras	XSSed	Tokenization Word2Vec	Cross-valid., Dropout	CNN-LSTIM	99.3	99.1	99.5	N/A
[17]		SLM	TensorFlow / Keras	Private	Tokenization	Dropout, Cross-valid.	MLP	99.32	98.0	98.7	0.31
[18]	XSS-DOM	SLM	TensorFlow / Keras	Private	CBOW	Cross-valid.	DFFN	N/A	95.0	N/A	N/A
[19]	XSS in **PHP*** and JS	SLM	N/A	Private (D1, **D2***)	Wod2Vec, **Code2Vec***	Data cleaning, Data balancing	DFFN with attention	95.38	99.90	91.80	N/A
[20]	XPath	SLM	PyBRAIN	Private	Character embedding	Data cleaning, Boosting	RNN, LSTM	N/A	84.2	N/A	16.2
[21]	Phishing	SLM	N/A	UCI	Feature selection and weighting using GA.	Dropout, Boosting, Cross-valid.	DNN	88.77	85.81	N/A	N/A
[22]		SLM	N/A	UCI	Neural embedding	Data balancing,	DNN	97.71	90.51	92.16	1.7
[23]	DDoS	USLM	WEKA	Private	Statistical feature extraction	Cross-valid.	SAE, LR	N/A	98.99	N/A	1.27
[24]		SLM	N/A	CICIDS2017, **KDDCU99***	Neural embedding	Data balancing	MC-CNN	99.18	N/A	N/A	N/A
[25]	Webshell in **PHP***, JSP and ASP	SLM	TensorFlow	Private	CBOW	Dropout	CNN	99.50	99.70	99.40	N/A
[26]	Trojan	USLM	N/A	Private	Neural embedding	Denoising	SAE	94.92	N/A	N/A	16.32

* best results

Logistic regression (LG) [23] was used with SAE for classification, as it can output a value corresponding to the probability of belonging to a given class. Transformer architecture like BERT [15] showed promising results to be used in Web attack detection. BERT can be considered as a stack of encoders and decoders. To combine all the strengths of all models, a typical approach is to use an ensemble. Voting ensembles are primarily selected based on the output of models. Hard voting predicts the majority class, whereas soft voting summed predicted probabilities from individual classifiers for each class. So, soft voting gives better insight better if all models provide probabilities [10].

E. Over/under-fitting protection

Classifying imbalanced datasets is a major challenge. DL algorithms solve these challenges and thus do not need a feature selection process performed during preprocessing [5]. However, if irrelevant or unnecessary features are selected, high variance (overfitting) or high bias (underfitting) may occur. Summary of reviewed papers showed that data cleaning and/or balancing was the most used method to combat over/under-fitting. Other techniques were dropout, cross-validation, boosting, denoising, early stopping and data augmentation. Data cleaning [3, 10, 12, 14, 19, 20] includes operation such as removing duplicates, modifying labels without values, etc. Data balancing [4, 11, 12, 15 19, 22, 24] involves random sampling and random parameter initialization in order to obtain balanced data set. Dropout [7, 10, 16, 17, 21, 25] is usually applied only to the neurons in the top one to three layers (excluding the output layer). Grid or random search [16, 17, 21, 23] has been used as a model hyperparameter optimization technique with a k-fold cross-validation approach. This approach randomly splits the training dataset into a set of k-folds of approximately equal sub-dataset sizes. Different boosting techniques for updating the weights of the DL models were used, inducing Adam [10, 11, 13], RMSprop [20], and AdaDelta [19]. Denoising [7,24] works by corrupting the original input with some noise. The Autoencoder then needs to reconstruct the input from that noise, which forces the hidden layer to capture the statistical dependencies between the inputs. Another way to regularize iterative learning algorithms, such as gradient descent, is early stopping [8]. The training is stopped as soon as the validation error reaches a desired minimum. Finally, data augmentation [5] artificially increases the size of the training set by generating many modified variants of the training instance. In this way, it compensates for a shortage in data.

F. Evaluation

In imbalanced classification, the number of examples in the training dataset for each class label is not balanced. Since the distribution of classes is skewed, e.g., they extend much farther to the left or right of the median. In this case, it is not sufficient to use accuracy alone as a metric of model performance. So, it is recommended to examine the confusion matrix. The idea of a confusion matrix is to count the times instances of class A are classified as class B for all A/B pairs. Elements of the confusion matrix in the context of Web security are the following: true positives (TP) or Web attack requests in the dataset that are correctly detected as Web attacks, true negative (TN) or normal requests correctly detected as normal request, false positives (FP) are those normal requests detected as Web attack requests and false negatives (FN) that are Web attack requests detected as normal requests. To compute the confusion matrix, we first need to have a set of predictions to compare them to the actual targets. Each row in a confusion matrix represents an actual class, while each column represents a predicted class. A perfect classifier would only have true positives and true negatives, so its confusion matrix would only have nonzero values on its main diagonal. Analyzing the confusion matrix often gives us insight into ways to improve your classifier. Below are just a few of the most common performance metrics [17]:

$$Accuracy = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}} \quad (1)$$

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \quad (2)$$

$$Sensitivity/Recall/TPR = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \quad (3)$$

$$FPR = \frac{\mathrm{FP}}{\mathrm{TN+FP}} \quad (4)$$

$$F1 - score = 2\left(\frac{\mathrm{Recall} * \mathrm{Precision}}{\mathrm{Recall} + \mathrm{Precision}}\right) \quad (5)$$

$$AUC = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \quad (6)$$

The recall/precision trade-off should be noted. In this trade-off, the higher the recall is, the lower the precision becomes and vice versa. Because of this, the F1-score measure or the harmonic mean of precision and recall is usually applied [22]. AUC is an abbreviation for area under the ROC Curve, AUC is the probability that the model ranks a random positive example more highly than a random negative example. ROC stands for receive operating characteristic curve, and in a nutshell, it pots TPR vs FPR at different classification thresholds. A perfect classifier will have a ROC AUC equal to 1, whereas a purely random classifier will have a ROC AUC equal to 0.5. When a feature distribution has a heavy tail (i.e., when values from the mean are not exponentially rare), both min-max scaling and standardization will squash most values into a small range. So, before we scale the features, it is recommended first to transform it to shrink the heavy tail and, if possible, to make the distribution roughly symmetrical and bell-shaped distributions (e.g., by computing their logarithm or square root). To improve performance of the classifier, first we can gather more training data so that the classifier learns to distinguish them. Second, we could engineer new features that would help the classifier. Third, we can try to preprocess the data.

V. CONCLUSION

Client-side vulnerabilities exploit Web browser weaknesses. Most Web applications do not accurately filter the user's input data, so both the client and the server are invaded. The main difference between DL and traditional ML is that performance of DL algorithms benefits more when the dataset augments. DL models do not require complicated artificial feature engineering.

Retraining using online data opens the possibility of incorporating attack data into the normal dataset and avoiding detection. Class imbalance can cause DL models to be biased towards the negative class and negatively impact classification performance. Random sampling can help to balance it. The goal of detecting unseen attacks (zero-day) should be elaborated as detecting new attacks that share specific characteristics with the known attack used in training. Some models are better at limited dataset size and for specific Web attacks. Unsupervised DL models, like Autoencoders, can achieve high F1-scores in Web attack detection without domain knowledge and labeled training data. Publicly available datasets are outdated and not representative in most use cases. Therefore, we can conclude that data quality and feature extraction are critical components for the robustness and precision of these classifiers. Moreover, the proposed DL methods effectively detect Web attacks with FPR close to zero and the corresponding TPR (AUC-ROC curve) of almost 100%. Furthermore, codes and preprocessing details or datasets need to be publicly available for reproducibility and as a contribution to similar works in the future. New models should aim to achieve better detection accuracy, low computational cost, high flexibility, and high robustness. Some possible future works or improvements in this area are as follows: (i) finding and correcting misclassified labels in the original dataset, (ii) extracting more characteristic information of the Web attack besides the HTTP request or URL. (iii) creating a new standardized dataset of Web attacks (iv) improve existing approaches even further by exploring alternative combinations of DL models, (v) determine the frequency when DL models should be retrained and (vi) develop a methodology for creating new DL models and a framework for evaluating existing ones.

REFERENCES

[1] Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. applied sciences, 9(20), 4396.

[2] Ahmad, R., Alsmadi, I., Alhamdani, W., & Tawalbeh, L. A. (2023). Zero-day attack detection: a systematic literature review. Artificial Intelligence Review, 1-79.

[3] Gong, X., Lu, J., Wang, Y., Qiu, H., He, R., & Qiu, M. (2019, December). CECoR-Net: A character-level neural network model for web attack detection. In 2019 IEEE International Conference on Smart Cloud (SmartCloud) (pp. 98-103). IEEE.

[4] Liang, J., Zhao, W., & Ye, W. (2017, December). Anomaly-based web attack detection: a deep learning approach. In Proceedings of the 2017 VI International Conference on Network, Communication and Computing (pp. 80-85).

[5] Karacan, H., & Sevri, M. (2021). A novel data augmentation technique and deep learning model for Web application security. IEEE Access, 9, 150781-150797.

[6] Yu, L., Chen, L., Dong, J., Li, M., Liu, L., Zhao, B., & Zhang, C. (2020, July). Detecting malicious web requests using an enhanced textcnn. In 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 768-777). IEEE.

[7] Tian, Z., Luo, C., Qiu, J., Du, X., & Guizani, M. (2019). A distributed deep learning system for web attack detection on edge devices. IEEE Transactions on Industrial Informatics, 16(3), 1963-1971.

[8] Yang, W., Zuo, W., & Cui, B. (2019). Detecting malicious URLs via a keyword-based convolutional gated-recurrent-unit neural network. IEEE Access, 7, 29891-29900.

[9] Pan, Y., Sun, F., Teng, Z., White, J., Schmidt, D. C., Staples, J., & Krause, L. (2019). Detecting web attacks with end-to-end deep learning. Journal of Internet Services and Applications, 10(1), 1-22.

[10] Luo, C., Tan, Z., Min, G., Gan, J., Shi, W., & Tian, Z. (2020). A novel web attack detection system for internet of things via ensemble classification. IEEE Transactions on Industrial Informatics, 17(8), 5810-5818.

[11] Liu, T., Qi, Y., Shi, L., & Yan, J. (2019, August). Locate-Then-Detect: Real-time Web Attack Detection via Attention-based Deep Neural Networks. In IJCAI (pp. 4725-4731).

[12] Tang, R., Yang, Z., Li, Z., Meng, W., Wang, H., Li, Q., ... & Liu, Y. (2020, July). Zerowall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications (pp. 2479-2488). IEEE.

[13] Tang, P., Qiu, W., Huang, Z., Lian, H., & Liu, G. (2020). Detection of SQL injection based on artificial neural network. Knowledge-Based Systems, 190, 105528.

[14] Xie, X., Ren, C., Fu, Y., Xu, J., & Guo, J. (2019). Sql injection detection for web applications based on elastic-pooling cnn. IEEE Access, 7, 151475-151481.

[15] Seyyar, Y. E., Yavuz, A. G., & Ünver, H. M. (2022). An attack detection framework based on BERT and deep learning. IEEE Access, 10, 68633-68644.

[16] Kadhim, R., & Gaata, M. (2020). A hybrid of CNN and LSTM methods for securing web application against cross-site scripting attack. Indones. J. Electr. Eng. Comput. Sci, 21, 1022-1029.

[17] Mokbal, F. M. M., Dan, W., Imran, A., Jiuchuan, L., Akhtar, F., & Xiaoxi, W. (2019). MLPXSS: an integrated XSS-based attack detection scheme in web applications using multilayer perceptron technique. IEEE Access, 7, 100567-100580.

[18] Melicher, W., Fung, C., Bauer, L., & Jia, L. (2021, April). Towards a lightweight, hybrid approach for detecting dom xss vulnerabilities with machine learning. In Proceedings of the Web Conference 2021 (pp. 2684-2695).

[19] Maurel, H., Vidal, S., & Rezk, T. (2022). Statically identifying XSS using deep learning. Science of Computer Programming, 219, 102810.

[20] Deshpande, G., & Kulkarni, S. (2019). Modeling and mitigation of XPath injection attacks for web services using modular neural networks. In Recent findings in intelligent computing techniques (pp. 301-310). Springer, Singapore.

[21] Ali, W., & Ahmed, A. A. (2019). Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm ‐ based feature selection and weighting. IET Information Security, 13(6), 659-669.

[22] Feng, F., Zhou, Q., Shen, Z., Yang, X., Han, L., & Wang, J. (2018). The application of a novel neural network in the detection of phishing websites. Journal of Ambient Intelligence and Humanized Computing, 1-15.

[23] Yadav, S., & Subramanian, S. (2016, March). Detection of Application Layer DDoS attack by feature learning using Stacked AutoEncoder. In 2016 international conference on computational techniques in information and communication technologies (icctict) (pp. 361-366). IEEE.

[24] Chen, J., Yang, Y. T., Hu, K. K., Zheng, H. B., & Wang, Z. (2019, February). DAD-MCNN: DDoS attack detection via multi-channel CNN. In Proceedings of the 2019 11th International Conference on Machine Learning and Computing (pp. 484-488).

[25] Lv, Z. H., Yan, H. B., & Mei, R. (2019). Automatic and accurate detection of webshell based on convolutional neural network. In Cyber Security: 15th International Annual Conference, CNCERT 2018, Beijing, China, August 14–16, 2018, Revised Selected Papers 15 (pp. 73-85). Springer Singapore.

[26] Xuan, S., Man, D., Wang, W., Qin, K., & Yang, W. (2018, November). Hybrid Classification of WEB Trojan Exploiting Small Volume of Labeled Data Vectors. In 2018 14th International Conference on Computational Intelligence and Security (CIS) (pp. 286-290). IEEE.