Computers, Materials & Continua
DOI:10.32604/cmc.2022.020389
Article

Tech Science Press

MNN-XSS: Modular Neural Network Based Approach for XSS Attack Detection

Ahmed Abdullah Alqarni[1], Nizar Alsharif[1], Nayeem Ahmad Khan[1,*], Lilia Georgieva[2], Eric Pardade[3] and Mohammed Y. Alzahrani[1]

[1]Department of Computer Sciences and Information Technology, AlBaha University, AlBaha, Saudi Arabia
[2]Department of Computer Science, Heriot-Watt University, Edinburgh, UK
[3]Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia
[*]Corresponding Author: Nayeem Ahmad Khan. Email: nayeem@bu.edu.sa
Received: 22 May 2021; Accepted: 23 June 2021

Abstract: The rapid growth and uptake of network-based communication technologies have made cybersecurity a significant challenge as the number of cyber-attacks is also increasing. A number of detection systems are used in an attempt to detect known attacks using signatures in network traffic. In recent years, researchers have used different machine learning methods to detect network attacks without relying on those signatures. The methods generally have a high false-positive rate which is not adequate for an industry-ready intrusion detection product. In this study, we propose and implement a new method that relies on a modular deep neural network for reducing the false positive rate in the XSS attack detection system. Experiments were performed using a dataset consists of 1000 malicious and 10000 benign sample. The model uses 50 features selected by using Pearson correlation method and will be used in the detection and preventions of XSS attacks. The results obtained from the experiments depict improvement in the detection accuracy as high as 99.96% compared to other approaches.

Keywords: Cybersecurity; XSS; deep learning; modular neural network

1 Introduction

The number of web services is growing exponentially. Web applications which are accessed via web browsers have become primary targets for cybercriminals [1,2]. A report published by Symantec Corporation in 2019 implies that 439 million pieces of new varieties of malware were identified [3], With the distinct nature and behavior among the cybercriminals and cybersecurity solution provides to defend attacks thus making it very is challenging for cybersecurity defenders to discover in what manner the new kind of malware will appear [4]. The objective of conducting such types of attacks by cybercriminals is for individual, monetary and political benefits. Therefore, early detection of such malware attacks has emerged as the uppermost cybersecurity challenge.

The Open Web Applications Security Project (OWASP) has declared Cross-site scripting (XSS) as one of the top vulnerabilities which are exploited to perform attacks [5]. XSS attacks are malicious script code attacks which are injected into malicious or legitimate and trusted websites [6]. These malicious scripts are delivered illegitimately to the user's machine so that vulnerabilities are exploited in web applications, and attacks are performed. Mostly these malicious payloads are delivered to users through email attachments or on visiting a compromised website. Cybercriminals commit a breach on the legitimate but vulnerable website to infuse the malicious script inside or develop a phishing website. Poor programming practices which such as not covering all the security aspects is one of the significant causes of vulnerability in a web application [7]. Malicious JavaScript is often employed to perform XSS attacks. JavaScript being a scripting language has several advantages which include adding versatility, dynamism, interactive into the webpages. A significant advantage of using JavaScript is that it reduces the computation load on the server-side but executing the scripts on users' side through a web browser [8]. JavaScript offers several advantages; however, the downside is that it provides a solid foundation to conduct the XSS attack. These attacks are performed on the users' side, which is executed by the web browsers, but there is no mechanism in web browsers for detecting any malicious scripts. Web browsers run all the scripts sent by the server, whether malicious or benign. Execution of malicious JavaScript by web browser may lead to user session hijacking, manipulating the legitimate website, for example by injecting malicious code or content or phishing attacks [9].

Deep learning as a new field of research is a subset of machine learning and works on by imitating forming of connections in a human brain modelled as neural networks, and also been applied to detect malware [10]. With the Neural Network's advancement, the problems of previous machine learning approaches in terms of accuracy in malware detection have increased. The likelihood of enhanced classification accuracy appears by developing a neural network with a higher number of prospect layers, also known as deep learning. Preliminary studies in deep learning that have been employed to detect malware in Android mobiles confirm that malware was detected with high accuracy [11]. In this study, we propose using network-based neural networks to detect malicious XSS code attacks.

The motivation of using the Deep Neural Network (DNN) for the detection of XSS attacks is to remove the necessity of domain expertise in feature extraction, remove complexity and solve the problem end to end. A modular neural network works on the concept of implementing multiple individual neural networks. These neural networks are trained instantaneously for a particular subtask and the results achieved are combined to perform the single task.

The major contribution of this study is the widespread use of the Word2vec model for the detection of malicious JavaScript's using MNN. Our meticulous experiments using the Word2vec model and MNN reveals a much better performance. None of the existing studies has successfully employed Word2Vec and MNN to show this degree of performance. Hence determine that our method is an effective method for detecting malicious JavaScript attacks.

The rest of the paper is organized as: Section 2 gives the details about related work. In Section 3 the overview of the proposed approach is detailed. In Section 4 experimental details and evaluation are presented. Section 5 concludes the work.

2 Related Work

Existing solutions [12,13] for detection of malicious code attacks are broadly based on two approaches: signature-based and heuristic-based. In the signature-based approach, the malicious

script's detection is performed by comparing the unique string patters in the binary code. The unique strings are created from previously captured instances of malicious code. A security solution based on signature-based needs frequently updates of their database with new signatures [14]. There is a huge time gap between finding the new malware variant and updating the signature of the malware into the database on the client-side. The attackers take benefit of such time gap to launch the attack and may affect millions of devices. Signature-based approach for malware detection fails in such an environment where new malware variants are expected to arrive.

Another approach used for the detection of malware is heuristic-based detection [15]. In a heuristic-based approach, the detection is performed using an expert system based on expert decision rules. Based on the set criterion, the expert system will decide whether a piece of code is suspicious or benign. The major downside of this approach is the long scanning time deciding whether a code is malicious or benign. Another challenge with this approach is that it has a high false-positive rate. To overcome the challenges in signature-based and heuristic-based detection approaches such as of high false-positive rate, long scanning time, frequent updating of signatures at the client-side, researchers use machine learning.

Several alternative approaches have been proposed to detect malicious JavaScript attacks using machine learning and non-machine learning methods. In this section, the machine learning and deep learning-based approaches for detecting malicious code attacks are reviewed related to our approach. As a study by [16], proposed and implemented an approach for the detection of malicious code-based N-gram, and the classification was performed using SVM. N-gram was used to generate N tokens consecutively in a stream for feature extraction. The experiments were conducted using 1831 instances of SQL injection and XSS. The experimental result shows that precision of 98.04% was obtained with a true false positive of 0.985% and a false positive rate of 0.015% when used on trigram. The downside of this approach is that the tokenizers need constant training to detect malicious code. A study by [17], used machine learning classifiers such as SVM, Naïve Bayes, J48 and bagging for the classification of malicious and benign code. The features extracted from the user-input context were used along with some basic features particularly related to input, output, validation, and sanitization routines. Experimental results show that an accuracy of 92.6% was achieved using bagging. Shar et al. [18] proposed a predication model for detection on XSS vulnerabilities based on machine learning classification and clustering techniques. Hybrid attributes extracted using static and dynamic analysis were used for code for vulnerability prediction. The experiment was performed on six applications, and results show that an average of 90% recall and 85% precision was obtained. The downside of the approach is that it has huge performance overheads and high false positive rate. A study by Fang et al. [19] presented an approach for detecting XSS using deep learning. In this study the features were extracted from XSS payload. They used Long Short-Term Memory (LSTM) recurrent neural network for detection. Experimental results show that precision of 99.5% was achieved. The downside of this approach is that it has a high time complexity. A study by Stokes et al. [20] proposed and implemented a deep recurrent neural for the detection of malicious JavaScript's. A hybrid of static and dynamic analysis was used. The presented model is highly complex, and the results produced are not convincing. Experimental results show that the applied LaMP model achieved a 65.9% true positive rate, and the best CPoLS model obtained 45.3% true positive rate, with 1.0% as a false positive rate.

3 Proposed Approach

3.1 Overview

Our new proposed approach for the detection of malicious JavaScript is based on deep and modular neural network. The proposed approach works on a self-learning method capable of detecting known and unknown variants of malware. The property of using a system which is self-learning and utilised machine learning and deep learning models that enables to extract the convoluted features from the code snippets to differentiate between the benign and malware code. The processes involved in this detection approach is shown in Fig. 1.

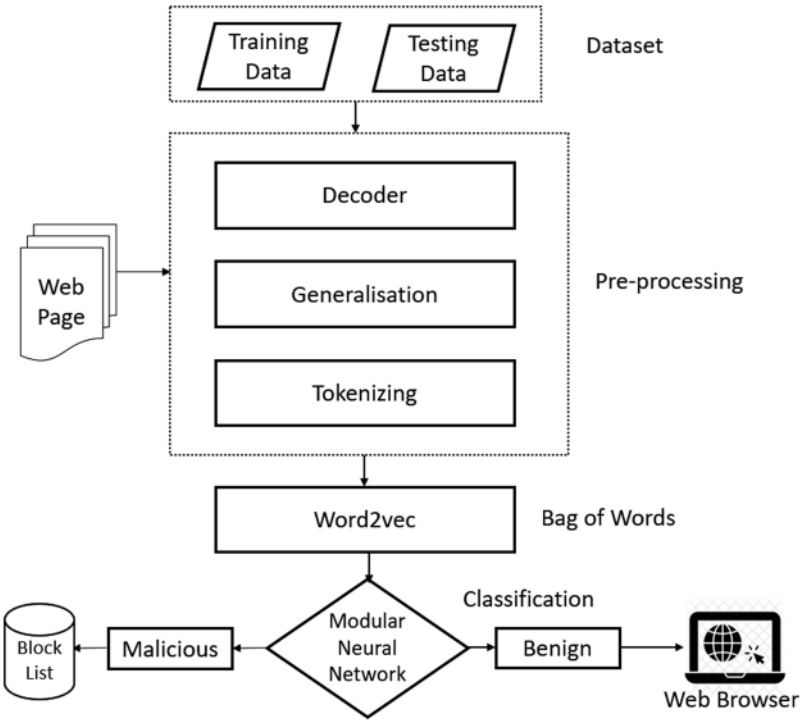

Figure 1: Architecture of the proposed approach

3.2 Pre-Processing

The pre-processing in this approach includes decoding, generalisation and tokenizing. The first step towards detecting malicious XSS is performing decoding on the code segment, which needs to be tested for malicious or benign. The attacker's use obfuscation techniques to evade the detection bypassing the traditional filters and validation mechanisms. The obfuscation or encoding is done through techniques such as Unicode, Hex encoding Base64, UTF-7 encoding. Using all the possibilities, in this proposed approach the decoder will decode data and will bring the code to a normal format. The second step in pre-processing is generalisation. This step involves removing the data noise, meaningless and non-helpful information from the decoded code, and the from the normal code. The generalisation includes removing black spaces, special characters, http://, and conversion of function parameters to param_strings. The third step is performing the tokenizing on the data. The purpose of using tokenization here is to break the sequence of strings into pieces involving input characters, sub-characters, or subgroups. Another benefit of using tokenization

is that it minimizes the length of data and reduces the complexity, leading to lessens the data handling cost. In tokenization only important word remain therefore increasing the accuracy.

3.3 Word2vec and CBOW Model

In this approach, we have considered each code instances as a plain text and treating it like a natural language. For this purpose, we use word2vec [21]. Word2vec algorithm employs a neural network model to learn associations of words from the large text corpus. Once the model is trained; it will help detect the synonymous words and recommend some extra words. As the name indicates, word2vec shows every distinct word with a specific list of numbers towards a vector. The vectors are selected so that a simple mathematical cosine function will depict the words represented by those vectors. Continuous bag of words (CBOW) and continuous skip gram are two models which word2vec can use to generate distributed representations of words. CBOW architecture model is usually considered much faster than skip gram for construct word representation. Keeping in view of the efficiency, In this study we employed CBOW model. Fig. 2 [22] shows depicts the architecture of CBOW.

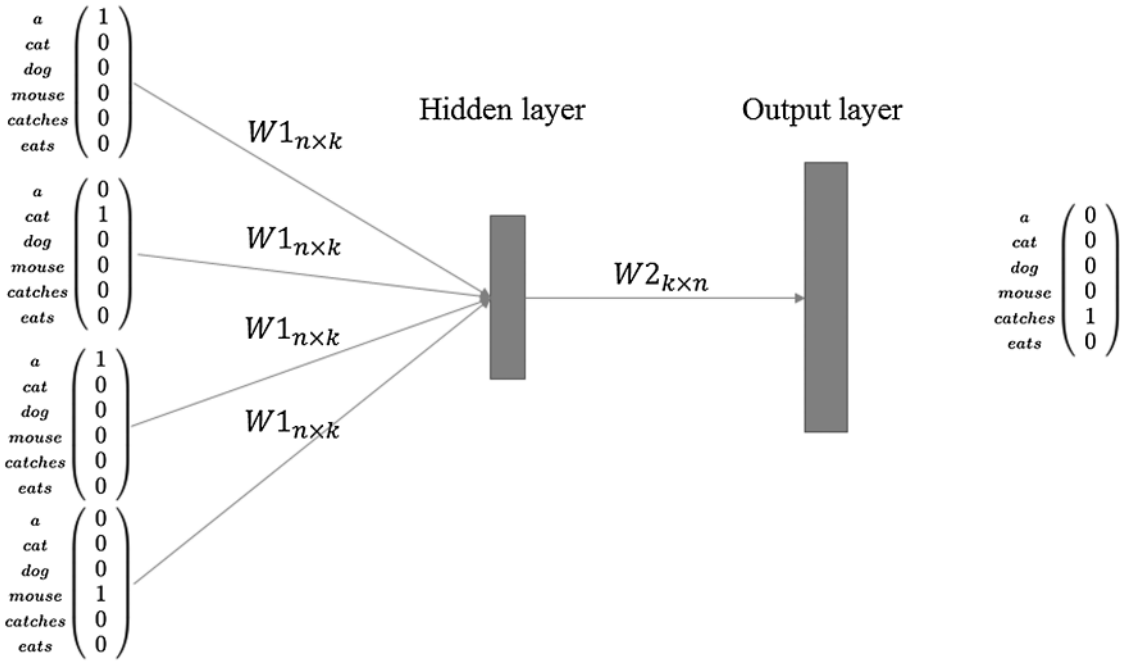

Figure 2: Architecture of CBOW model [22]

3.4 Feature Selection

Feature selection is a method in which the number of the features are reduced as input variables for generating a predictive model [23]. The objective of feature selection is to reduce the computational cost and performance overheads and enhance a model's prediction accuracy. Suppose we have a feature vector F, we have to find the most optimal feature set F', keeping in mind that not all the features will contribute to the prediction model's accuracy.

Given a set of features $F = \{f\_(1,)f\_(2)\ldots f\_(n\ )$, find the subset $F' \subseteq F$ which will maximise the learner's ability to classify patterns.

In this study we used Pearson's Correlation method which is a filter-based selection method [24]. Pearson's Correlation method is useful in determining the association among the continuous features and the class [25]. The mathematical representation is shown in Eq. (1).

$$r = \left(\sum \frac{[(x-\bar{x})(y-\bar{y})]}{\sqrt{\sum (x-\bar{x})^2 \sum (y-\bar{y})^2}} \right) \tag{1}$$

Only top n features are selected in our dataset by determining the absolute value of correlation among the target and numerical features. The high association was calculated by using 'CorrelationAttributeEval' package in WEKA [26]. The selected features are given in Tab. 1.

Table 1: Selected feature list

Feature number	Feature name	Feature number	Feature name
1	url_length	26	html_attr_profile
2	url_special_characters	27	html_attr_http-equiv
3	url_tag_script	28	html_event_onblur
4	url_attr_src	29	html_event_onchange
5	url_event_onload	30	html_event_onclick
6	url_event_onmouseover	31	html_event_onerror
7	url_cookie	32	html_number_keywords_evil
8	url_number_keywords_param	33	js_file
9	url_number_domain	34	js_pseudo_protocol
10	html_tag_script	35	js_dom_location
11	html_tag_iframe	36	js_dom_document
12	html_tag_meta	37	js_prop_cookie
13	html_tag_object	38	js_method_write
14	html_tag_embed	39	js_method_getElementById
15	html_tag_link	40	js_method_alert
16	html_tag_svg	41	js_method_eval
17	html_tag_frame	42	js_method_fromCharCode
18	html_tag_div	43	js_min_length
19	html_tag_style	44	js_min_define_function
20	html_tag_img	45	js_min_function_calls
21	html_tag_input	46	js_string_max_length
22	html_attr_classid	47	html_length
23	html_attr_codebase	48	js_method_getElementsByTagName
24	html_attr_href	49	js_prop_referrer
25	html_attr_longdesc	50	html_event_onmouseup

3.5 Deep and Modular Neural Network

The last step in this detection approach is using modular neural network (MNN) to detect malicious JavaScript's. MNN is considered one of the most influential and independent artificial neural networks, which is changed with only a few intermediate values [27]. MNN are neural networks which symbolize the ideas and principles of modularity. The property of using the

modularity is that it can be broken down into several generally free, replicable, and composite modules. MNN reduces the computational complexity and enhances system performance and robustness [28]. The results obtained have been highly desirable than monolithic system which is based on a rigid structure. The input features are analysed by MNN, which further breaks down features into sub-features and each network is processed independently. During the process, the output generated from individual networks are consumed by the intermediary process as an input to generate the final output. The intermediary process has a characteristic of taking each process individually and perform the required action without getting distracted from other signals and doesn't interrelate without other networks. Basically, the strategy used by MNN to solve the problem is based on "divide and conquer method" [29]. MNN divides the highly complex task into a multiple subtask and each subtask is handled individually by each module. The solution produced from subtasks are combined through a unified multi module decision making strategy. Keeping in view of the advantages of using modular network, in this study optimised neural network is implemented for the detection of malicious JavaScript code attacks. Fig. 3 [30], depicts the basic structure of MNN and considered as a collection of monolithic neural networks that each deal with a subset of a problem and then have their separate outputs merged by an integration unit to form a comprehensive solution to the entire issue. The basic principle is that a complex problem can be broken down into simpler subsets that simpler neural networks can solve. The entire solution can be a blend of the outputs of the simple monolithic neural networks.

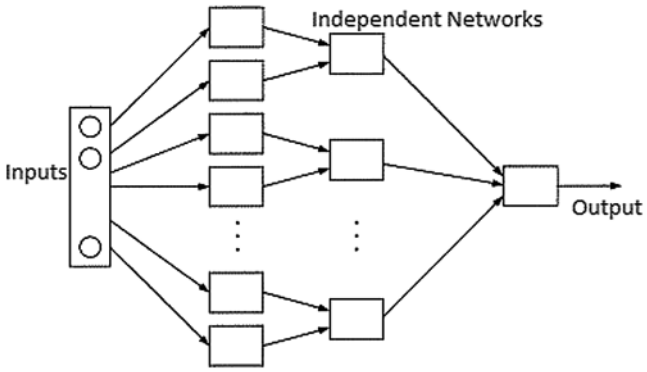

Figure 3: Basic structure of MNN [30]

The output "O" generated by each independent network is combined to produce the final optimised output and is mathematically represented as given in Eq. (2). The presence or absence of the module is known through the coefficient of the network module.

$$O = \frac{\sum_{i=1}^{n} a_i^2 g_i k_i}{\sum_{i=1}^{n} g_i} \tag{2}$$

where, $a_i^2 =$ Module output in which $i \in [1, 2, \ldots n]$, $g_i =$ Average deviation of the generated output by module i, $k_i =$ Coefficient of module i.

In this proposed approach for XSS detection using MNN, each module in neural network takes as its input from the dataset. Each module in this study a 2-layer multilayer preceptor where the output generated by the second layer on the neural network is a_i^2 $(for\ i \in (1, \ldots, n))$, where n is the maximum number of modules. In this study we are using 2 modules as given in

Fig. 4 and module integration is done using Eq. (2). The reason of selecting only 2 modules is to simplify calculations. The output generated by modules is given as input to module integrator for making the final decision. The integrator works based on a threshold for deciding whether a code is malicious or benign. If the module integrator generated an output greater than 0.5 the code is classified as malicious, and if it is less than 0.5 the code is benign as shown in Fig. 5. The efficiency of the proposed approach is evaluated using experimental results.

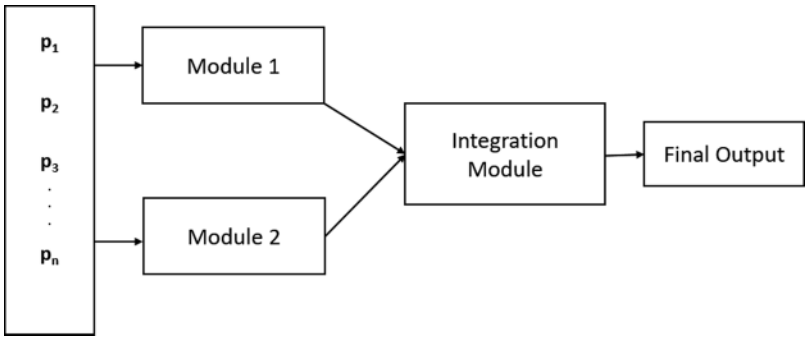

Figure 4: Modular neural network with 2 modules

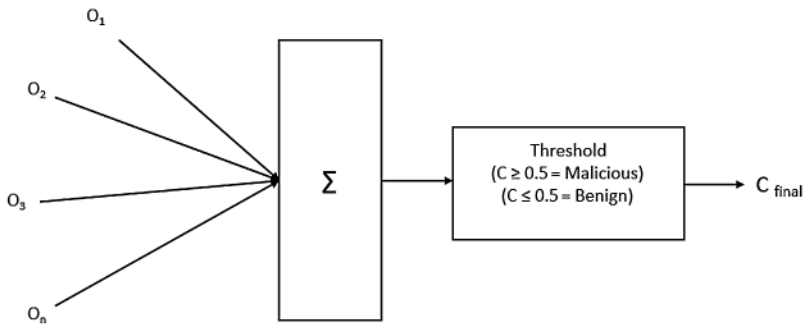

Figure 5: Working of module integrator

4 Experimental Setup

4.1 Dataset

The dataset used in this study was obtained from figshare.com [31] developed by authors [32]. The dataset consists of 101000 instances with 1000 as malicious and 100000 benign instances. The dataset contains 67 features based on three categories viz HTML, JavaScript, and URL. The sample class is represented by [0, 1], 0 for benign and 1 for malicious benign.

4.2 Experimental Environment & Evaluation

The experiments to confirm the effectiveness of this approach was conducting on i5, 3.5 Ghz, 8 GB RAM. MATLAB platform was used to develop and experiment modular neural network. WEKA was used for the feature selection based on Pearson correlation. In first the dataset obtained was divided into 80% training and 20% testing data. Further 10-fold cross validation scheme was used for resampling of the data and generate predictions from unseen data. In this

study only 50 features were selected based on Pearson correlation among the total of 68 features as given in the main dataset. The features selected are given in Tab. 1. To compare the results generated from proposed modular neural network we used two approach: Backpropagation Neural Network (BPNN) and Radial Basis Function Network (RBFN) on the same data using same parameters. The only change was the number of features selected. in both approach we used 25 features each. The first 25 features were used in Backpropagation Neural Networking and second 25 features were used in and Radial Basis Function Network. The hidden layer in our experiments was set 20 and the activation function used were "tansig" and "purelin". The training was continued till 500 epochs to achieve best accuracy result without any additional performance overheads. The confusion matric was used as for basic evaluation. Accuracy, Precision, Recall and F-1 score was measured in this study. These evaluation metrics are widely known and accepted by the research community. The formula for calculating each metrics is given in Eqs. (3)–(5) respectively. The results obtained are given in Tab. 2.

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

$$Precison = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F1 = 2 * \frac{Precison * Recall}{Precison + Recall} \tag{6}$$

where, TP = True Positive, TN = True Negative, FP = False Positive, FN = false Negative.

Table 2: Experimental results and comparison with other methods

Metrices	Methods					
	BPNN		RBFN		Proposed MNN	
	Benign	Malicious	Benign	Malicious	Benign	Malicious
Accuracy (%)	98.66		97.34		99.96	
Precision (%)	96.95	100	95	100	99.95	99.95
Recall (%)	100	96.38	100	92.0	100	99.91
F1-score (%)	98.32	97	96.45	95.67	99.95	99.92

Based on the insight obtained from the experimental results it is evident that our proposed MNN achieved an accuracy of 99.96% which was high compared to BPNN and RBFN which achieved 98.66% and 97.34% respectively using all the features mentioned in Tab. 1. An increase of 1% in any detection approach means hundreds of attackers can be detected. Thus, the proposed MNN-XSS approach has proven to be highly effective in detecting XSS attacks.

5 Conclusion

In this study, we propose MNN for XSS detection and conducted experiments. The experimental results show that the proposed approach achieved an accuracy of 99.96% in detecting novel malicious JavaScript based XSS attack upon learning. The selection of number on neurons is one of the main paraments that are required to be specified the realization of the MNN. This further leads to decrease in complexity and thus allows implementation of neural network with limited resources. The Pearson correlation played an important role in feature selection thus lead to the higher accuracy.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] N. Khan, J. Abdullah and S. A. Khan, "Defending malicious script attacks using machine learning classifiers," *Wireless Communications and Mobile Computing,* vol. 17, pp. 1–9, 2017.

[2] M. Yar, and K. F. Steinmetz, "Cybercrime and the Internet," in *Cybercrime and Society,* 3rd Edition, London, UK, SAGE, 2019. [Online]. Available: http://www.books.google.com.

[3] "Symantec corporation annual report-2019," 2019. [Online]. Available: https://docs.broadcom.com/doc/istr-24-2019-en.

[4] T. J. Holt and M. G. Turner, "Examining risks and protective factors of on-line identity theft," *Deviant Behaviour,* vol. 33, pp. 308–323, 2012.

[5] "Open Web Application Security Project report 2020: Top 10 Web application security risks, 2020," 2020. [Online]. Available: https://owasp.org/www-project-top-ten/.

[6] N. A. Khan, J. Abdullah and A. S. Khan, "Towards vulnerability prevention model for web browser using interceptor approach," in *9th Int. Conf. on IT in Asia, (IEEE-CITA-15)*, Kuching, Malaysia, pp. 1–5, 2015.

[7] N. A. Khan, J. Abdullah and A. S. Khan, "A dynamic method of detecting malicious scripts using classifiers," *Advance Science Letters,* vol. 23, no. 7, pp. 5352–5355, 2017.

[8] P. Laskov and N. Srndic, "Static detection of malicious JavaScript-bearing PDF documents," in *27th Annual Computer Security Applications Conf.*, Florida, USA, pp. 373–382, 2011.

[9] P. Likarish, E. Jung and I. Jo, "Obfuscated malicious JavaScript detection using classification techniques," in *4th IEEE Int. Conf. on Malicious and Unwanted Software (MALWARE)*, Montreal, Quebec, Canada, pp. 47–54, 2009.

[10] N. A. Khan. M. Y. Alzahrani and H. A. Kar, "Hybrid feature classification approach for malicious javaScript attack detection using deep learning," *International Journal of Computer Science and Information Security,* vol. 18, no. 5, pp. 79–90, 2020.

[11] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *Journal of Systems Architecture,* vol. 112, no. 101861, pp. 1–24, 2020.

[12] K. Gupta, R. Singh and M. Dixit, "Cross site scripting (XSS) attack detection using intrusion detection system," in *Int. Conf. on Intelligent Computing Systems*, Madurai, India, pp. 199–203, 2017.

[13] G. Canfora, A. Sorbo, F. Mercaldo and C. A. Visaggio, "Obfuscation techniques against signature-based detection," in *Proc. of Mobile Systems Technologies Workshop*, Milan, Italy, pp. 21–26, 2015.

[14] C. M. R. Silva, E. L. Feitosa and V. C. Garcia, "Heuristic-based strategy for phishing prediction: A survey of URL-based approach," *Computers & Security,* vol. 88, no. 101613, pp. 1–19, 2020.

[15] D. Zhang, X. Han and C. Deng, "Review on the research and practice of deep learning and reinforcement learning in smart grids," *Journal of Power and Energy Systems,* vol. 4, no. 3, pp. 362–370, 2018.

[16] J. Choi, H. Kim, C. Choi and P. Kim, "Efficient malicious code detection using n-gram analysis and SVM," in *14th Int. Conf. on Network-Based Information Systems*, Tirana, Albania, pp. 618–21, 2011.
[17] M. K. Gupta, M. C. Govil and G. Singh, "Predicting cross-cite scripting (XSS) security vulnerabilities in web applications," in *12th Int. Joint Conf. on Computer Science and Software Engineering*, USA, pp. 162–167, 2015.
[18] L. K. Shar, H. B. K. Tan and L. C. Briand, "Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis," in *35th Int. Conf. on Software Engineering*, San Francisco, USA, pp. 642–651, 2013.
[19] Y. Fang, Y. Li, L. Liu and C. Huang, "DeepXSS: Cross site scripting detection based on deep learning," in *Int. Conf. on Computing and Artificial Intelligence*, Chengdu, China, pp. 47–51, 2018.
[20] J. W. Stokes, R. Agrawal and G. McDonald, "Neural classification of malicious scripts: A study with JavaScript and vbscript," arXiv preprint arXiv: 1805.05603, 2018.
[21] K. W. Church, "Word2vec," *Natural Language Engineering,* vol. 23, no. 1, pp. 155–162, 2017.
[22] G. Bouala, "Word embedding models: Word2vec, Camembert and USE," *Le Blog de Baamtu,* 2020.
[23] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research,* vol. 3, pp. 1157–1182, 2003.
[24] "Spss tutorials: Pearson correlation," Kent University Library Tutorial, 2021. [Online]. Available: https://libguies.lides.library.kent.edu/SPSS/PearsonCorr.
[25] E. C. Blessie and E. Karthikeyan, "Sigmis: A feature selection algorithm using correlation based method," *Journal of Algorithms & Computational Technology,* vol. 6, no. 3, pp. 385–394, 2012.
[26] "Class correlationAttributeEval," WEKA, University of Waikato, New Zealand, 2020. [Online]. Available: https://weka.sourceforge.io/doc.dev/weka/attributeSelection/CorrelationAttributeEval.html.
[27] V. Golovko, S. Bezobrazov, P. Kachurka and L. Vaitsekhovich, "Neural network and artificial immune systems for malware and network intrusion detection," *Advances in Machine Learning,* vol. 2, pp. 485–513, 2010.
[28] G. Kourakos and A. Mantoglou, "Pumping optimization of coastal aquifers based on evolutionary algorithms and surrogate modular neural network models," *Advances in Water Resources,* vol. 32, no. 4, pp. 507–521, 2009.
[29] U. Lahiri, A. Pradhan and S. Mkhopadhyaya, "Modular neural network-based directional relay for transmission line protection," *IEEE Transactions on Power Systems,* vol. 20, no. 4, pp. 2154–2155, 2005.
[30] A. Abugabah, A. AlZubi, F. Al-Obeidat, A. Alarifi and A. Alwadain, "Data mining techniques for analyzing healthcare conditions of urban space-person lung meta-heuristic optimized neural networks," *Cluster Computing,* vol. 23, pp. 1781–1794, 2020.
[31] F. M. M. Mokbal, D. Wang, X. Wang and L. Fu, "Data augmentation-based conditional wasserstein generative adversarial network-gradient penalty for XSS attack detection system," *PeerJ Computer Science,* vol. 6, no. e328, pp. 1–20, 2020.
[32] F. Makbal, "Cross-ite scripting attack (XSS) dataset," *Figshare*. 2021. [Online]. Available: https://figshare.com/articles/dataset/XSS_dataset1_csv/13046138?file=24959207.