JKAU: Comp. IT. Sci., **Vol**. **13** No. 1, pp: 68 – 84 (2024 A.D.)
DOI: 10.4197/Comp.13-1.4

Employing Sequence to Sequence Neural Network Model for XSS Attack Detection

Mohammad Eid Alzahrani[1]

[1]*Department of Computer Science, Al-Baha University*
Al-Baha, KSA meid@bu.edu.sa

Abstract. Cross-site scripting (XSS) attacks are considered one of the most prevalent types of attacks and have caused huge damage to individuals and organizations in the form of economic loss and intrusion into privacy. Several detection techniques have been used to find known threats using signatures obtained from network traffic. Researchers have developed many techniques based on machine learning to identify attacks without depending on known signatures of already known attacks. While a number of neural network-based methods to detect XSS attacks have been proposed by security experts, no one has attempted to detect XSS attacks using a sequence neural network model. We have proposed a novel approach called a sequence- to-sequence neural network (seq2seq) model to detect cross-site scripting attacks without depending on signatures of known attacks. Using seq2seq model for XSS detection is based on extracting features from web application code segments, then using them to predict whether a script contains malicious code. The seq2seq model is represented as a two-layer neural network, with the first layer processing the training samples in sequential order and the second layer responsible for the classification of each data sample. This dataset consists of 10100 instances of malicious and benign JavaScript. The Pearson correlation method was used for feature selection. All the experiments were conducted using Tenserflow and Keras. The experimental results that proposed seq2seq achieved an accuracy of 99.8%

Keywords—Cybersecurity, XSS Attacks, Deep Learning.

I. INTRODUCTION

Numerous security challenges in web applications have been caused by the Internet's fast proliferation and the complex func- tionality of web applications. New attacks have emerged target-
ing the interplay between web applications and their underlying databases [1]. Developers need to learn about the potential se- curity problems associated with the different technologies. Poor programming methods may cause certain weaknesses, while mali- cious scripting by the attackers behind the scenes can cause other issues [2]. Attackers are continually developing methods to get access to confidential information using online apps. By accessing user data or seizing control of system resources, applications that are open to hostile users might undermine the system's security and protection measures. Once exploited, the systems may be used to launch further attacks against other machines on the network. The most common known attacks are SQL injection, which exploits the poor quality of programming methods; Cross-site scripting (XSS), a malicious approach that targets different types of apps and sites; Cross-site request forgery (CSRF), which has the same goals as XSS; Session hijacking, which tries to steal data from an already initiated session, and Shell command injection (SCI), which aims
to seize control of the machine through a backdoor shell [3]. These attacks can

undermine network security and the integrity of busi- ness data if they are carried out successfully. Thus, in applications that expose data to the Internet or open networks and make use of insecure programming methods, a defense strategy should first focus on assuring that malicious attacks cannot gain access to user data and system resources. As per Open Web Application Secu- rity (OWASP) year 2021 report (A03:2021-Injection), cross-site scripting attacks are number three among the most reported vul- nerabilities [4]. In this attack, the attacker injects a malicious script into the page, which the victim executes. The vulnerability occurs when user input is not properly validated before being used in a dynamic script that is included on the page. When validat- ing untrusted data, the application should verify that dangerous characters are properly encoded or escaped and reject known bad inputs. The developer should always sanitize user inputs by en- coding or escaping HTML tags and JavaScript code to prevent this attack. Deep learning (DL), a subfield of machine learning (ML) [5], consists of layers of an artificial neural network (ANN) that are inspired by the neuronal structure of the human brain, with part of the neurons in each layer having activation functions that pro- vide non-linear outputs. Deep learning has been applied to various fields, from natural language processing to image recognition [6].

The deep underlying architecture of neural networks used in DL has also been widely replicated and utilized. They can be trained using various machine learning methods, such as supervised, un- supervised, and reinforcement learning. The limitations of earlier machine learning techniques in terms of accuracy in malware de- tection have improved with the development of neural networks [7]. By creating a deep-learning neural network with a larger number of prospect layers, classification accuracy can be improved. Furthermore, deep learning models often work faster than machine learning models, particularly for complex tasks or problems with large amounts of data. Keeping in view the advantages offered by deep learning, we propose employing a sequence-to-sequence neural network for detecting XSS attacks. This approach involves training a neural network to first detect cross-site scripting (XSS) injection points in an application and then learn the specific pat- terns of malicious input. The contribution of this paper can be summarized as:

- First, this research study's primary goal is to employ the seq2seq approach to find XSS attacks in web applications.
- Second, the Proposed seq2seq approach seeks to detect XSS attacks in real time.
- Third, to find an optimal feature set using the feature selection method.
- Forth, to achieve higher detection accuracy using different model parameters.

II. WORKING OF XSS ATTACKS

XSS is an application-level flaw vulnerability affecting mil- lions of users. To exploit this flaw, the attacker can plant code on a Web page that subverts how users see and interact with an application when they click on a malicious link or enter data into a tampered form [2]. Cross-site scripting flaws often provide cyber- criminals a way to pretend to be a victim user, execute whatever operations they can do, and get hold of any of the victim's data. The attacker may be able to fully manage all of the functionality and data of the program if the target user has privileged access to it. XSS allows attackers to steal information from user accounts, read and modify data sent from the server, and impersonate a victim. XSS is malicious JavaScript found in web applications that allow attackers to hijack an authenticated user's session [8]. To trick a website into returning dangerous JavaScript to visitors, XSS is used. The attacker may

absolutely obstruct the victim's engagement with the application when the malicious code runs within the victim's browser by injecting a payload that forces the browser to run some arbitrary JavaScript. JavaScript is often executed in a fairly regulated environment by web browsers. The operating system and files of the user are only partially accessible to JavaScript. However, JavaScript may still be harmful if used improperly as part of malicious material. The user's cookies may also be accessed using Javascript. Session tokens are often kept
in cookies. A user's session cookie gives an attacker access to the user's personal information and allows them to operate in the user's place while impersonating them. JavaScript has the ability to read the Document Object Model (DOM) of the browser and make arbitrary changes to it. Fortunately, this only applies to the page on which JavaScript is active. The XMLHttpRequest object in JavaScript may be used to send HTTP requests with any content to any destinations. Modern browsers that support HTML5 APIs can utilize JavaScript [9]. It may be able to access the user's geolocation, camera, microphone, and even certain files from their file system, for instance. The majority of these APIs demand user opt-in. However, the attacker may get past this restriction by using social engineering. OWASP has divided XSS attacks into three categories: Reflected, Stored and DOM-based XSS attacks.

1) Reflected XSS

Reflected-XSS is an attack where the attacker injects mali- cious code into a website that the user reflects. The website will then send this malicious code back to the victim, who will execute it in their browser [10]. Attackers often use reflected-XSS attacks as a way to steal users' cookies. They do this using JavaScript or VBScript, which can be injected into websites and executed when users visit them [11].

2) Stored XSS

This type of XSS occurs when the malicious code is stored in the database and not executed until it is retrieved. This type of attack is more difficult to detect because there are no obvious signs that a script has been injected into the database [12].

3) DOM-based XSS

DOM-based XSS exploits the DOM in the browser. DOM is an independent and language-neutral platform that permits applications and scripts to dynamically access and update web pages, or any other document, on the fly. The DOM spec- ification defines how to represent objects in HTML, XML, and other formats in a system-independent manner. A DOM- based XSS vulnerability occurs when user input is echoed into an HTML page without proper validation or escaping; this can be triggered simply by loading a page with an attacker's URL parameter containing malicious script code [13].

III. RELATED WORK

Several solutions have been developed to detect malicious JavaScript in the context of suspicious websites, with numerous studies successfully detecting malicious JavaScript by using ma- chine learning to scan malicious JavaScript. In this study, our focus is on the approaches based on machine learning and deep learning. A study by [14] proposed an approach for detecting ma- licious code in the source code of web pages. The features were extracted from URL and JavaScript code snippets. The classifi- cation was done using three machine learning classifiers: SVM,
Nave Bayes and J48 Decision Trees. According to an investigation by [15], a method for identifying malicious code-based N-grams was developed and implemented utilizing SVM for classification. An N-gram technique was used to generate tokens from the code. The experiment yielded an accuracy of 98.04%, a precision rate of 0.98% , and a recall rate of

0.015% when trigram was applied for tokenizing. The disadvantage of this strategy is that the tokenizers need continuous training to identify malicious code. A study by [16] developed an approach for predicting XSS vulnerabilities us- ing classification and clustering methods of machine learning. In this study, the authors used hybrid features extracted by using static and dynamic analysis. The approach performed well in terms of precision and recall, but the limitation of this study is that it has huge performance overheads, which makes it difficult to use in real-time detection. The overall performance of machine learning- based approaches for XSS was very good, but deep learning-based approaches were developed to improve the accuracy further. Real- time detection with high accuracy is crucial in critical applications. A study by [17] used the LSTM deep learning approach for XSS detection. Based on this approach, the features are automatically extracted from the work vector using the Word2Vec method. An- other study by [18] proposed an approach based on modular neural networks for detecting XSS. The authors used 50 features selected from a real-life dataset. The studies mentioned in this study will be used to compare the results obtained from our proposed approach to measure the effectiveness of the approach.

IV. PROPOSED APPROACH

The XSS-based attack happens due to security flaws in websites and owing to characteristics, and web browsers support sophisti- cated functionalities that come from dynamic online applications. Although these features are appealing and practical, they pose se- rious threats and raise web applications' security vulnerabilities. Cybercriminals often take advantage of these weaknesses. This study proposes a novel seq2aeq Neural Network Model for XSS attack detection based on this problem. The model is depicted in Figure 1. This detection model comprises three main components: data collection, feature selection and representation, and seq2seq neural network model. The architecture of the proposed approach is depicted in Figure 1. This approach has not been explored be- fore and it's worth exploring as it could provide new insights into how malware detection can be improved. The novelty of using seq2aeq Neural Network Model for malware detection is that it does not require any human intervention and is able to detect new threats without any training on the specific types of malware to be detected.

A Dataset

The dataset utilized in this work was compiled by authors [19] and downloaded from [20]. The details of the dataset are given in Table I.

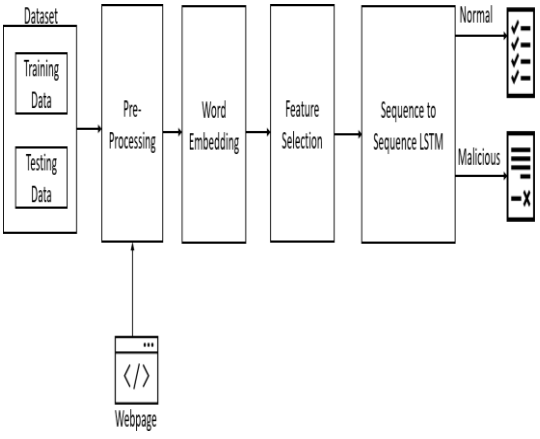

Fig. 1. Proposed Approach

Table I. Dataset Details

Name of Dataset	XSS dataset1
Format of Dataset	Format of Dataset
Size of Dataset	14.65 MB
Number of Instances	10100
Category of Instances	3
Number of Features	50
Number of Classes	2

B Preprocessing

This technique comprises decoding, generalization, and tok- enization in its preprocessing. The first step in detecting dangerous XSS is decoding the code segment to determine if it is malicious or benign. The attackers utilize obfuscation methods to circum- vent typical filters and validation processes to avoid detection. In this suggested method, the decoder will use all options to decode data and convert the code to a standard format. Generalization is the second phase of preprocessing [21]. This stage eliminates data noise, irrelevant and useless information, and the normal code from the decoded code.

C Feature Selection

Feature selection is an important technique in machine learn- ing. It is used to identify the best subset of features that can be used to train a model to achieve the best performance in an application [22]. The selected features should be as relevant for the training of a model as possible and not misleading. A good feature selection technique is useful to reduce overfitting or reduce variance in a model's performance [23]. In many cases, domain experts have already selected some features based on their domain knowledge. Feature selection is important in machine learning as it helps to reduce overfitting, including reducing variance in a model's per- formance. A good feature selection technique is useful to reduce overfitting and can help with decision-making by domain experts. When selecting features, one should take into account the proper- ties of each possible feature and the types of suitable models The main objective of feature selection is to improve the performance of a machine learning algorithm by reducing the number of features it needs to process and, when possible, by increasing the accuracy of predictions [24]. The feature selection process includes set- ting criteria for features and eliminating those that don't meet the definition. Preprocessing features that meet the definition and sum- marizing them into groups or clusters. Select the most important and informative features from each cluster to use them in machine

learning models [23]. Analyzing the performance of models that use the features selected in order to identify which features were most effective at predicting a target outcome, there are two main types of feature selection algorithms: filter-based and wrapper- based methods [25]. Filter-based methods filter out features based on their individual statistical significance, while wrapper-based methods use a statistical test or computational complexity mea- sure to evaluate all possible subsets of features and select the best subset. Wrapper-based methods have been shown to perform bet- ter in practice and are generally preferred. However, filter-based methods are best suited for researchers who do not need to perform any additional analysis on the features selected by a wrapper-

based method [26].

C.1Pearson Correlation

Pearson Correlation for feature selection is a statistical mea- sure of the linear association between two variables. It can be used to determine the strength and direction of a relationship be- tween different variables [27]. The Pearson correlation coefficient ranges from -1 to 1. A value of 1 means a perfect positive linear relationship between the variables; a value of -1 means a perfect negative linear relationship between the variables, while a value of 0 means no linear relationship between the two variables. The Pearson correlation coefficient is an important statistic in statistics and econometrics because it has desirable mathematical proper- ties that make it easy to compute from data sets containing large numbers of observations [28]. The Pearson correlation coefficient can be used to assess the degree to which one variable may help predict another variable, even if there is no causal relationship be- tween them. The relationship between the features is determined using the correlation approach. The most important one is the parametric correlation approach, which uses the sample data set to estimate the population parameters like mean, variance, skew- ness, and kurtosis [29]. There are two basic groups for measuring the correlation between two random variables. One is based on traditional linear correlation, while the other is founded on infor- mation theory. The most widely known of these two measures is the linear correlation coefficient, which is the covariance of two random variables, X and Y, divided by the product of their standard deviations.

The linear correlation 'r' for variables (X, Y) is shown as:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x - \bar{x})^2 \sum(y - \bar{y})} \quad (1)$$

are ranked according to the decreasing order. Then the feature- feature correlation is done to remove the redundant feature. While applying this feature selection to our dataset, only 30 features were used in this study.

D Term Frequency–Inverse Document Frequency

The importance of information retrieval for anti-malware soft- ware cannot be overstated [30]. Without solid filtering mecha- nisms, malware detection would probably not exist as we know it today. Information retrieval methods are frequently used for this purpose in anti-malware software. Term frequency-inverse document frequency (TFIDF) is one such method that has been successfully implemented for malware detection [31]. Term fre- quency (TF) measures how often a term appears in a document or collection of documents, while inverse document frequency (IDF) is a way to account for how common or rare the term is in general. The final TF-IDF value is generated by multiplying these numbers together and is given as:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \quad (2)$$

where, $tf(t, d)$ =Number of times term "t" appears in a document "d".

idf(t) = Inverse document frequency of the term t.

The relevance of a term is decided by its TF-IDF score. If the score is higher, the term is considered more significant or relevant compared to terms with low scores converging towards 0 [32]. TFIDF is a traditional method of creating numerical dataset vectorization. TFIDF is used to numerically represent a large collection of documents or data through their feature vectors which can then be used for classification and clustering. We develop a classification model from the sequential data

r = correlation coefficient,
to identify and remember important information in a sequence of data, such as identifying the parts of a sentence that are relevant
x_i = values of the x-variable in a sample,
$\bar{x}$ mean of the values of x-variable,
y_i = values of the y-variable in a sample,
$\bar{y}$ mean of the values of the y-variable
In Pearson correlation, the selected attributes are ordered in descending order, and the correlation between features eliminates the irrelevant and redundant features. All the selected features

for a specific task. A recurrent neural network has a sequence of recurrent layers that process the entire sequence simultaneously. LSTM networks have many layers that process parts of the se- quence at a time, with an LSTM unit inside each of these layers [34]. These LSTM units can learn to process long sequences of inputs, like video, and produce a single output representing the entire sequence at the end. LSTM can look back in time to see the previous states and gradients of the network better to determine

the current and future states of a network. There are three main reasons LSTM networks can do this: The LSTM network allows the network to remember data that has been seen analyzed using the LSTM technique in this study. The current text data must be digitized when a model uses different classification techniques. We employed the TF-IDF, the most used text digitization method, for this purpose.

E The LSTM Model

Long Short-Term Memory (LSTM) is a recurrent neural net- work (RNN) architecture designed to address the problem of van- ishing or exploding gradients in neural networks [33]. It can learn
in the past. It uses a ”gated” mechanism to decide if and when data from the past should be integrated into the current state of the network [35]. This mechanism uses two parallel paths: a ”future” path that predicts what the network will see in the future and a ”past” path that looks back at what the network has seen. The future path has a ”forget” gate determining when previous information should be thrown out. The past path has a ”remember” gate determining when previous information should be kept. The ”select” gate is a bias node that determines which path the network should use to make its decision at each time step. The architecture of LSTM is depicted in Figure 2.

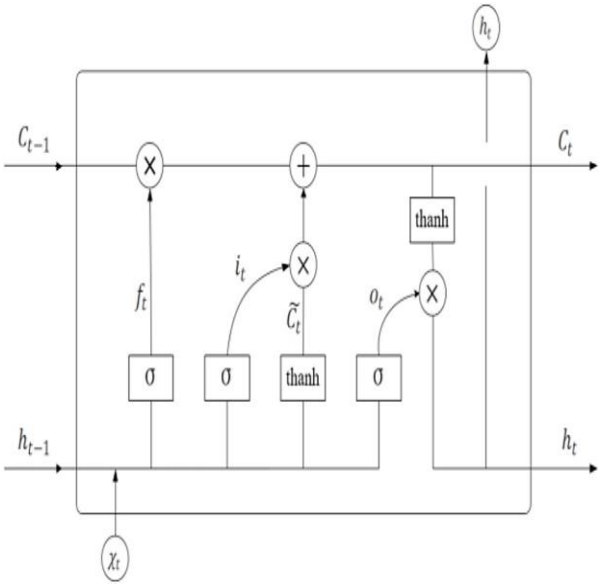

Fig. 2. LSTM Architecture

Mathematically LSTM is represented as:

$$i(t) = \sigma(W(i)x(t) + u(i)h(t-1)) \quad //inputgate \quad (3)$$

$$f(t) = \sigma(W)(f)x(t) + u(f)h(t-1) \quad //forgetgate \quad (4)$$

$$o(t) = \sigma(W(o))x(t) + u(o)h(t-1)) \quad //output/exposuregate \quad (5)$$

$$c\tilde{}(t) = tanh(W(c)x(t)) + u(c)h(t-1) \quad //newmemorycell \quad (6)$$

$$c(t) = f(t)oc\tilde{}(t-1) + i(t)oc\tilde{}(t) \quad //latememorycells \quad (7)$$

$$h(t) = o(t)otanh(c(t)) \quad (8)$$

Mentioning the meanings of the parameters and variables in Fig- ure 3. Building an LSTM network requires attention to three main architectural choices: the number of layers, the size of each layer, and the type of each gate. The LSTM architecture consists of three types of layers: input, output, and hidden. The input and output layers contain a single layer of cells where each cell is connected to the cells in the same layer in the other sequence and to cells in the next layer. The hidden layers are composed of several LSTM

Parameters and variables	Meaning
σ	Sigmoid function, range from 0 to 1
tanh	Hyperbolic tangent function, range from -1 to 1
$*$	Element-wise multiplication
i_t	Input gate
f_t	Forgot gate
o_t	Output gate
c_t	State of current memory cell at time t
$\widetilde{c_t}$	Candidate value for state at time t
h_t	Output value
x_t	Input value
W_i, W_f, W_o, W_c	Weights
R_i, R_f, R_o, R_c	Weights
b_i, b_f, b_o, b_c	Bias vectors

Fig. 3. Meaning of the parameters and variables

layers inside each hidden layer, where each LSTM layer contains several LSTM cells [36]. Researchers have demonstrated the effectiveness of LSTM networks in malware detection. Malware often attempts to avoid detection by changing its signature over time, but LSTM networks can track subtle changes in malware behavior over months or years. In one example, researchers used an LSTM network to track the evolution of the CryptoLocker ransomware family: CryptoLocker remained largely unchanged during the first 12 months of operation, but new versions appeared every few months thereafter. This was a sign that the malware authors had learned about the strengths and weaknesses of the initial version and were making improvements. The LSTM network observed subtle changes in CryptoLocker's code that were not immediately apparent to humans [37].

F Sequence to Sequence Model

Sequence to Sequence (seq2seq) models, a special family of recurrent neural networks (RNN), have been used to address chal- lenging natural language processing issues [38]. The models have also been widely used in machine translation, summarization, and dialogue systems. This model can ingest long sequences of words or characters and answer questions that depend on the structure of the long input sequence. seq2seq models achieve good perfor- mance on many challenging tasks by using the context surrounding an input token to determine the most likely output token [39]. seq2seq models use attention mechanisms to determine which pieces of the input sequence are most relevant to the output to- ken they are attempting to predict. The architecture of the seq2seq model is shown in Figure 4 [40]. We address the long-term depen- dence issue for detecting XSS attacks in this research and provide a novel seq2seq model. Our seq2seq model analyses the source code of a website to look for XSS attacks. The fundamental concept is taken from seq2seq modeling, which uses an encoder-decoder architecture and is extensively utilized in voice recognition and lan- guage translation, where the seq2seq is successful in modeling the long-term dependence of the words. In our architecture, an LSTM network is used as the encoder to convert the sequence of code scripts into a latent feature space containing compressed vectors,

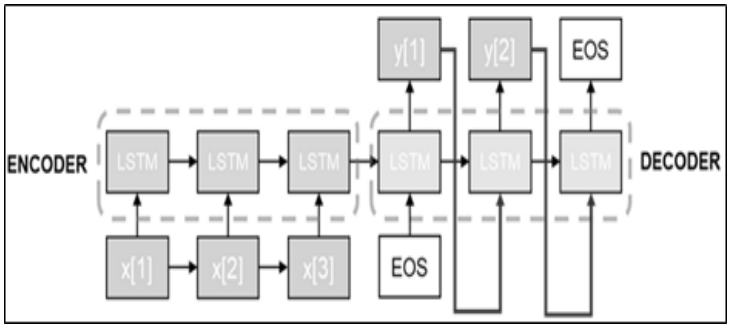

Fig. 4. The architecture of seq2seq Model

and a second LSTM is used as the decoder to convert the vectors into a series of potential attacking phases. This will allow us to completely eliminate the necessity for developing further stage de- tection and provide an end-to-end approach for sequence detection. Furthermore, since the LSTM serves as the fundamental building block for both our encoder and decoder, its advantages in terms of long-sequence learning provide us with a superior capacity to simulate the stages' long-term dependence.

Standard recurrent designs are inadequate for modeling inter- actions between sequences of differing lengths. To increase the flexibility of load forecasting, seq2seq, an alternative architecture based on LSTM, is utilized to map sequences of varying lengths. A possible solution is to use two LSTMs: an encoder LSTM that converts a series of inputs into a single hidden vector and a decoder LSTM that turns this hidden vector into a sequence of outputs [41]. The Seq2seq approach was developed to more efficiently manage input and output sequences of varying lengths. Each cell com- prises LSTM cells and two neural networks: an encoder for the input and a decoder for the output.

F.1 Encoder

The encoder handles individual tokens in the input sequence. It aims to pack all the information about the input sequence into a vector of fixed length i.e., the 'context vector'. While cycling through all the tokens, the encoder delivers this vector onto the decoder [42]. The decoder takes as input a context vector and a hidden state of the layer below and produces an output. It is now the turn of the decoder to follow a similar procedure but in a way reverse to that followed by the encoder.

F.2 Context vector

The vector is created in such a manner that it's anticipated to capture the complete meaning of the input sequence and assist the decoder in making correct predictions [42]. The context vector is initialized in a way that assists it in capturing the overall meaning of the input sequence and generating complete outputs. The context vector can be constructed from a weighted sum of previous hidden states and feature attention weights. This is, however, a dangerous and very complex method to use in a sequence-to-sequence setting since the potential number of combinations grows exponentially with the increase in the length of inputs. As it is stated earlier, these problems are solved by initializing the context vector with

a weighted sum of previous hidden states and feature attention weights. Thus, one method to create a context vector is to make a linear combination of a varying number of previous hidden

F.3 Decoder

The decoder is an LSTM whose beginning

states are initialized to the end states of the encoder LSTM, i.e., the context vector of the encoder's last cell is input to the first cell of the decoder network. Using these starting states, the decoder begins creating the output sequence, and these outputs are also considered for subsequent outputs. Through all this, information is lost, and connections are drawn between the starting states of the encoder LSTM and the final state of the decoder LSTM [43]. The Attentional Seq2Seq function is implemented by using an encoder RNN to learn representations of the input sequence and a decoder RNN to generate the output sequence from the encoder's hidden representations [44]. The attention mechanism enables the decoder to concentrate on various areas of the input at each time step in the output sequence; it is analogous to the alignment mechanism seen in conventional statistical translation models. Seq2Seqe models lack the implicit deterministic requirement of unidirectional connectionist temporal classification (CTC models); hence, they are more adaptable in terms of input-output reordering [45]. The framework uses a two-step pipeline to carry out translation; the first step models how a decoder can generate an output sequence given a hidden representation of the input, while the second step builds on this idea to decode outputs without directly observing. This scheme is motivated by a view of the decoder that sees it as a separate entity that independently generates outputs given the encoder's representations. We experimented with LSTM cells and evaluated the effect of using a bidirectional encoder for Seq2Seq models. We feed the encoder the input sequence in reverse order. TensorFlow's implementation with an embedding layer was employed in our study, In addition, it can automatically create training examples based on the tagged corpus. During training, an input sequence was consumed by a recurrent cell in the forward direction and read back as is when making predictions.

V. EVALUATION METHOD

The confusion matrix is a table that is used to evaluate the performance of machine learning algorithms. It has four values: true positives, false positives, true negatives and false negatives. F-score is one of the most popular evaluation metrics for machine learning. The F-score is a metric that evaluates how well an algo- rithm performs when classifying data. It measures the accuracy of an algorithm by calculating a ratio of true positives and true negatives. In order to find out the F-score for an algorithm, we need to know its precision and recall values first. Precision mea- sures how many times the algorithm correctly identified a positive sample from among all samples that it classified as positive, while recall measures how many times it correctly identified a positive sample from among all samples that it classified as either positive or negative.

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FlasePositive + FalseNegative}$$

train the seq2seq model following hyperparameters used are given in Table II.

Table II. Model Hyperparameters

TruePositive

1	Iteration performed	20000
2	LSTM cell numbers	32
3	LSTM layers	3
4	Learning Rate	0.4
5	Drop Rate	0.6

$$Precison = \frac{}{TruePositive + FalsePositive} \quad (10)$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (11)$$

$$F1 = 2 \times \frac{Precison \times Recall}{Precison + Recall}$$

VI. EXPERIMENTAL SETUP AND RESULTS

The experimental results obtained are given in Table III, and a comparison of our proposed seq2seq model with other existing deep learning-based approaches is depicted in Table IV.

This study's experiments were done using a PC with a processor of 4.20 GHz, an Intel core i5 with 8 GB RAM. Python, Tensorflow 'and Keras were used to perform the experiments. TensorFlow is a framework that allows to implementation and train deep learning models. It also supports a few different types of model archi- tectures to provide flexibility as well. Deep learning models can be conceptualized as multilayered graphs with nodes and edges. Nodes represent mathematical operations, and edges represent the flow of data between operations. The graph is formed with nodes and edges when the user writes model code. The framework sched- ules those nodes and edges and creates a multilayer graph. The graph is implemented in memory and executed when the user exe- cutes the model. When the user executes the model, the framework converts the graph from graph format to executable format. When the user creates a new model, the framework creates a new graph for that model. Then it creates a session for the user to interact with that graph. The session is used to add nodes and edges to the graph and also to execute them. Keras is a high-level neural networks API running on top of TensorFlow, PyTorch, or Theano. It allows users to build neural networks and run existing models. Keras provides a simple way to create and initialize a neural net- work. Using Keras allows one to choose the right architecture for the problem and easily make modifications as needed. The high-level API of Keras allows developers with limited experience in machine learning to build and experiment with neural networks easily. The simplicity of Keras also makes it easy to integrate into other models or applications that use TensorFlow as their nu- merical computation engine. An important part of the computer security industry is the detection of malicious executables. Much research is being done to develop new models that can be used to detect malicious executables. TensorFlow and Keras have been used to implement a seq2seq neural network that can be used to de- tect malicious XSS attacks. The model has been trained using a set of known malicious XSS instances and a set of

known benign XSS instances. This experiment used 10-fold cross-validation (CV) to evaluate the proposed performance. CV divides the whole dataset into 10 similar size subsets in which 9 subsets are used for training while one is used for model validation. Based on the feature se- lection method, only 30 features were used in the experiment. To

Table III. Results Obtained

Model	Accuracy	Precision	Recall	F-Measure
SVM	95.4	0.953	0.952	0.956
Na¨ıve Bayes	94.7	0.951	0.953	0.952
K-NN	96.2	0.965	0.961	0.964
LSTM	97.5	0.964	0.963	0.966
Proposed seq2seq	99.8	0.992	0.998	0.994

Table IV. Comparison with other Deep learning Based Approach

Study	Accuracy	Precision	Recall	F-Measure
DeepXSS [17]	97.34	0.95	0.95	0.995
Proposed seq2seq	99.8	0.992	0.9983	0.994

Table III shows that the deep learning-based approach per- formed much better than traditional machine learning models, in which the proposed model achieved 99.8% detection accuracy. Similarly, the proposed seq2seq moles outperformed existing deep learning methods, as shown in Table III. The parameters shown in Table II provided the best results.

VII. CONCLUSION

XSS is one of the major security challenges affecting web appli- cations. We proposed and experimented with a seq2seq model for detecting XSS attacks in this study. Results obtained from exper- iments show that the proposed approach obtained an accuracy of 99.8%, which is so far high compared to existing approaches based on deep learning. The proposed approach can detect obfuscated and new variants of malicious JavaScript. The Pearson correlation as feature selection and hyperparameter selection played a vital role in achieving high accuracy.

REFERENCES

[1] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, and A. Wahab, "A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions," *Electronics*, vol. 9, no. 7,
p. 1177, 2020.

[2] B. K. Ayeni, J. B. Sahalu, K. R. Adeyanju, *et al.*, "Detecting cross-site scripting in web applications using fuzzy inference system," *Journal of Computer Networks and Communica- tions*, vol. 2018, 2018.

[3] G. E. Rodr´ıguez, J. G. Torres, P. Flores, and D. E. Benavides, "Cross-site scripting (xss) attacks and mitigation: A survey," *Computer Networks*, vol. 166, p. 106960, 2020.

[4] T. OWASP, "Web application security risks. 2021," 10.

[5] M. Torres-Vela´zquez, W.-J. Chen, X. Li, and A. B. McMil- lan, "Application and construction of deep learning networks in medical imaging," *IEEE transactions on radiation and plasma medical sciences*, vol. 5, no. 2, pp. 137–159, 2020.

[6] M. Khan, B. Jan, H. Farman, J. Ahmad, H. Farman, and
Z. Jan, "Deep learning methods and applications," *Deep learning: convergence to big data analytics*, pp. 31–42, 2019.

[7] D.-L. Vu, T.-K. Nguyen, T. V. Nguyen, T. N. Nguyen, F. Mas- sacci, and P. H. Phung, "A convolutional transformation net- work for malware classification," in *2019 6th NAFOSTED conference on information and computer science (NICS)*, pp. 234–239, IEEE, 2019.

[8] T. Singh and Meenakshi, "Prevention of session hijacking using token and session id reset approach," *International Journal of Information Technology*, vol. 12, pp. 781–788, 2020.

[9] P. Papadopoulos, P. Ilia, M. Polychronakis, E. P. Markatos, S. Ioannidis, and G. Vasiliadis, "Master of web puppets: Abusing web browsers for persistent and stealthy computa- tion," *arXiv preprint arXiv:1810.00464*, 2018.

[10] S. A. Lakhapati, P. Shirbhate, S. Jagtap, and A. Shrirang, "Cross site scripting attack," *International Journal of Elec- tronics, Communication and Soft Computing Science & En- gineering (IJECSCSE)*, pp. 131–135, 2018.

[11] I. F. Khazal and M. A. Hussain, "Server side method to detect and prevent stored xss attack.," *Iraqi Journal for Electrical & Electronic Engineering*, vol. 17, no. 2, 2021.

[12] K. Anagandula and P. Zavarsky, "An analysis of effective- ness of black-box web application scanners in detection of stored sql injection and stored xss vulnerabilities," in *2020 3rd International Conference on Data Intelligence and Secu- rity (ICDIS)*, pp. 40–48, IEEE, 2020.

[13] P. Wang, J. Bangert, and C. Kern, "If it's not secure, it should not compile: Preventing dom-based xss in large-scale web development with api hardening," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 1360–1372, IEEE, 2021.

[14] B. Vishnu and K. Jevitha, "Prediction of cross-site scripting attack using machine learning algorithms," in *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, pp. 1–5, 2014.

[15] J. Choi, H. Kim, C. Choi, and P. Kim, "Efficient malicious code detection using n-gram analysis and svm," in *2011 14th International Conference on Network-Based Information Sys- tems*, pp. 618–621, IEEE, 2011.

[16] L. K. Shar, H. B. K. Tan, and L. C. Briand, "Mining sql injection and cross site scripting vulnerabilities using hybrid program analysis," in *2013 35th International Conference on Software Engineering (ICSE)*, pp. 642–651, IEEE, 2013.

[17] Y. Fang, Y. Li, L. Liu, and C. Huang, "Deepxss: Cross site scripting detection based on deep learning," in *Proceedings of the 2018 international conference on computing and artificial intelligence*, pp. 47–51, 2018.

[18] F. M. M. Mokbal, D. Wang, X. Wang, and L. Fu, "Data augmentation-based conditional wasserstein generative ad- versarial network-gradient penalty for xss attack detection system," *PeerJ Computer Science*, vol. 6, p. e328, 2020.

[19] F. Makbal, "Cross-ite scripting attack (xss) dataset," 2021.

[20] A. Bennatan, Y. Choukroun, and P. Kisilev, "Deep learning for decoding of linear codes-a syndrome-based approach," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1595–1599, IEEE, 2018.

[21] Y. Chen, Y. Li, X.-Q. Cheng, and L. Guo, "Survey and tax- onomy of feature selection algorithms in intrusion detection system," in *Information Security and Cryptology: Second SKLOIS Conference, Inscrypt 2006, Beijing, China, Novem- ber 29-December 1, 2006. Proceedings 2*, pp. 153–167, Springer, 2006.

[22] P. Saari, T. Eerola, and O. Lartillot, "Generalizability and simplicity as criteria in feature selection: Application to mood classification in music," *IEEE Transactions on audio, speech, and language processing*, vol. 19, no. 6, pp. 1802– 1812, 2010.

[24] A. Abraham, F. Pedregosa, M. Eickenberg, P. Gervais,
A. Mueller, J. Kossaifi, A. Gramfort, B. Thirion, and G. Varo- quaux, "Machine learning for neuroimaging with scikit- learn," *Frontiers in neuroinformatics*, vol. 8, p. 14, 2014.

[25] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Is- lam, and J. Yearwood, "Hybrids of support vector machine wrapper and filter based framework for malware detection," *Future Generation Computer Systems*, vol. 55, pp. 376–390, 2016.

[26] J. Brownlee, "How to choose a feature selection method for machine learning," *Machine Learning Mastery*, vol. 10, 2019.

[27] K. Rawal and A. Ahmad, "Feature selection for electrical demand forecasting and analysis of pearson coefficient," in *2021 IEEE 4th International Electrical and Energy Confer- ence (CIEEC)*, pp. 1–6, IEEE, 2021.

[28] H. Ko, B. Son, Y. Lee, H. Jang, and J. Lee, "The eco- nomic value of nft: Evidence from a portfolio analysis us- ing mean–variance framework," *Finance Research Letters*, vol. 47, p. 102784, 2022.

[29] L. Cen, C. S. Gates, L. Si, and N. Li, "A probabilistic discrim- inative model for android malware detection with decompiled source code," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 400–412, 2014.

[30] T. A. Le, T. H. Chu, Q. U. Nguyen, and X. H. Nguyen, "Malware detection using genetic programming," in *the 2014 Seventh IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pp. 1–6, IEEE, 2014.

[31] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, pp. 29–48, Cite- seer, 2003.

[23] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learn- ing," in *2014 science and information conference*, pp. 372– 378, IEEE, 2014.

[32] C. Gao, J. Yan, S. Zhou, P. K. Varshney, and H. Liu, "Long short-term memory-based deep recurrent neural networks for target tracking," *Information Sciences*, vol. 502, pp. 279–296, 2019.

[33] M. Gao, G. Shi, and S. Li, "Online prediction of ship behavior with automatic identification system sensor data using bidirectional long short-term memory recurrent neural network," *Sensors*, vol. 18, no. 12, p. 4211, 2018.

[34] G. Petneha´zi, "Recurrent neural networks for time series fore- casting," *arXiv preprint arXiv:1901.00069*, 2019.

[35] X. Luo and L. O. Oyedele, "Forecasting building energy consumption: Adaptive long-short term memory neural net- works driven by genetic algorithm," *Advanced Engineering Informatics*, vol. 50, p. 101357, 2021.

[36] B. Zhang, W. Xiao, X. Xiao, A. K. Sangaiah, W. Zhang, and J. Zhang, "Ransomware classification using patch-based cnn and self-attention network on embedded n-grams of opcodes," *Future Generation Computer Systems*, vol. 110,
pp. 708–720, 2020.

[37] M. Aouad, H. Hajj, K. Shaban, R. A. Jabr, and W. El-Hajj, "A cnn-sequence-to-sequence network with attention for residential short-term load forecasting," *Electric Power Systems Research*, vol. 211, p. 108152, 2022.

[38] S. Hao, D.-H. Lee, and D. Zhao, "Sequence to sequence learning with attention mechanism for short-term passenger flow prediction in large-scale metro system,"

Transportation Research Part C: Emerging Technologies, vol. 107, pp. 287– 300, 2019.

[39] W. Kim, Y. Han, K. J. Kim, and K.-W. Song, "Electricity load forecasting using advanced feature selection and optimal deep learning model for the variable refrigerant flow systems," *Energy Reports*, vol. 6, pp. 2604–2618, 2020.

[40] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer net- works," in *Advances in Neural Information Processing Sys- tems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[41] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[42] Y. Yang, J. Zhou, J. Ai, Y. Bin, A. Hanjalic, H. T. Shen, and Y. Ji, "Video captioning by adversarial lstm," *IEEE Transac- tions on Image Processing*, vol. 27, no. 11, pp. 5600–5611, 2018.

[43] C. Zhou, L. Chen, J. Liu, X. Xiao, J. Su, S. Guo, and H. Wu, "Exploring contextual word-level style relevance for unsu- pervised style transfer," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020.

[44] T. Nakatani, "Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration," in *proc. INTERSPEECH*, vol. 2019, pp. 1408–1412, 2019.

[45] "Tensorflow availabe at: https://www.tensorflow.org/."

توظيف نموذج الشبكة العصبية (تسلسل لتسلسل) للكشف عن هجوم البرمجة النصية للمواقع

محمد عيد الزهراني[1]

[1] *قسم علوم الحاسبات، كلية الحاسبات والمعلومات*

جامعة الباحة، الباحة، المملكة العربية السعودية

meid@bu.edu.sa

مستخلص. تعتبر هجمات البرمجة النصية عبر المواقع واحدة من أكثر أنواع الهجمات انتشارا وقد تسببت في أضرار جسيمة للأفراد والمنظمات في شكل خسارة اقتصادية وانتهاك للخصوصية. تم استخدام العديد من تقنيات الكشف للعثور على التهديدات المعروفة باستخدام التوقيعات التي تم الحصول عليها من حركة مرور الشبكة. لقد طور الباحثون العديد من التقنيات القائمة على التعلم الآلي لتحديد الهجمات دون الاعتماد على التوقيعات المعروفة مسبقاً للهجمات المعروفة بالفعل. في حين تم اقتراح عدد من الأساليب القائمة على الشبكات العصبية للكشف عن هجمات البرمجة النصية عبر المواقع من قبل خبراء أمن المعلومات، لم يحاول أحد اكتشاف هجمات البرمجة النصية عبر المواقع باستخدام نموذج الشبكة العصبية التسلسلية. لقد اقترحنا نهجاً جديداً يسمى نموذج الشبكة العصبية من تسلسل إلى تسلسل لاكتشاف هجمات البرمجة النصية عبر المواقع دون الاعتماد على توقيعات الهجمات المعروفة. يعتمد استخدام نموذج تسلسل إلى تسلسل لاكتشاف هجمات البرمجة النصية للمواقع على استخراج الميزات من مقاطع التعليمات البرمجية لتطبيق الويب، ثم استخدامها للتنبؤ إذا كان البرنامج النصي يحتوي على تعليمات برمجية ضارة. يتم تمثيل نموذج تسلسل إلى تسلسل كشبكة عصبية ذات طبقتين، حيث تقوم الطبقة الأولى بمعالجة عينات التدريب بترتيب تسلسلي والطبقة الثانية مسؤولة عن تصنيف كل عينة بيانات. تتكون مجموعة البيانات هذه من ١٠١٠٠ نسخة من جافا سكريبت الضارة وغير الضارة. تم استخدام طريقة ارتباط بيرسون لاختيار الميزة. أجريت جميع التجارب باستخدام تنسرفلووكيراس. حققت النتائج التجريبية التي اقترحت تسلسل إلى تسلسل دقة قدرها تسعة وتسعون وثمانية من العشرة في المئة.

الكلمات المفتاحيةـ الامن السيبراني، هجمات البرمجة النصية عبر الواقع، التعلم العميق