European Journal of Computer Science and Information Technology
Vol.10, No.4, pp.1-13, 2022
Print ISSN: 2054-0957 (Print),
Online ISSN: 2054-0965 (Online)

A Robust System for Detecting and Preventing Payloads Attacks on Web-Applications Using Recurrent Neural Network (RNN)

O. E. Taylor[1] & P. S. Ezekiel[2]

[1,2]Department of Computer Science, Rivers State University, Port Harcourt, Nigeria

Citation: Taylor O. E. and Ezekiel P. S. (2022) A Robust System for Detecting and Preventing Payloads Attacks on Web-Applications Using Recurrent Neural Network (RNN), *European Journal of Computer Science and Information Technology*, Vol.10, No.4, pp.1-13

ABSTRACT- *Due to the growing rate of Internet usage, web apps have become the most popular Internet application. This has made web applications a significant objective for cyber-criminals; thereby, carrying various attacks on web applications like Cross-Site Scripting (XSS), Structured Query Language Injection and Shell attacks. Because of the high rate of web-based assault, this paper presents a robust framework for detecting and preventing multiple payload attacks on web applications. In this paper, an RNN model was trained on a dataset that contains different categories of assaults that are carried out on web applications. These attacks include: XSS, SQLi, and Shell. Random Over Sampling approach was used to resolve the problem of highly imbalanced dataset which prepared the dataset for preprocessing. After solving the imbalanced problem, pre-processing was then carried out on the dataset by performing data cleaning and tokenization. The tokenized data was transformed into an array which was used in feeding our RNN model as input. Our proposed model was trained on two (2) epochs, where each of the epochs shows the accuracy and loss values obtained by the model for both training and testing data. After training, our proposed RNN model gave us an accuracy of 99.96% for testing data and 99.91% for training data. We also deployed our RNN model to the web by making use of a python flask to build a robust system for detecting and preventing different payload attacks on web applications. This paper is limited to web-application attacks.*

KEYWORDS: payload attacks, recurrent neural network, web-applications, python flask

INTRODUCTION

With the advancement of cyber-space, web apps are playing a more vital role in people's daily lives than ever before. In the meantime, web applications have become particularly appealing attack targets. Attackers can steal users' private information, mess with the database, and destroy data by conducting attacks such as SQL injection and cross-site scripting (XSS). Because of the constant evolution of attack strategies, frameworks for detecting anomalies are utilized. to alert dangerous behaviours to maintain the safety of web applications. The web-attacks are mostly performed in the application layer payload, and the payload content is difficult and variable.

Algorithms for machine learning are commonly used by researchers to create payload anomaly, detection models. In order to train the detection models, they are split into unsupervised and supervised categories. Although unsupervised models do not require labeled data for training, they do require data that has already been labeled; trained models might have a high proportion of false positives. Models that have been supervised and trained on labeled data perform better [1].

Because of the expanding rate of Internet usage, web apps have become the most popular Internet application. Organisations can use web apps to boost revenue and enhance business operations, such as in the supply chain by using virtualization or a business platform. The "remain connected" characteristic of the Internet has revolutionized business life considerably. It allows users to contact others wherever and at any time. Information technologies and mobile cloud computing are all examples of Internet-based computing organizations that has benefited greatly from the Internet. Although application services do not require computational resources to run, different services such as computer software, storage, and servers are delivered to users' Personal Computers (PCs) or devices through the Internet via Web apps [2].

Because most businesses offer public Web services, they are dangerous to cyber-attacks. Cross-Site Scripting (XSS) and SQL Injection (SQLi), two of the most reported threats against Web applications, account for a major share of these assaults, in accordance with Wireless Application Service Provider (WASP). Successful assaults allow attackers to get more-than-necessary access to Web resources such as databases, resulting in severe sensitive data leakage or loss [3].

Web apps are vulnerable to cyber-attacks due to their widespread availability and high usage. By making a malicious request, attackers attempt to make the Web application inaccessible [4]. Attackers might corrupt a susceptible Web application, jeopardizing the organization's resources' confidentiality, integrity, and availability. Organizations may suffer financial losses and irreversible harm. Web applications can be harmed in different approaches, including damaging specific resources, stealing data from databases, disrupting operations, or gaining access to the Web application [5]. Web application security concerns include SQL injection, cross-site scripting (XSS), server-side inclusion, and broken authentication as specified by the Open Web Application Security Project's (OWASP)Top 10 Security Vulnerabilities 2013 report.

Related Works

[6] presented a deep learning anomaly-based Web attack detection architecture in a Web application. Data preprocessing and Convolution Neural Network (CNN) processes make up the architectural structure. CSIC2010v2 datasets were utilized to demonstrate the applicability and success of the proposed CNN architecture. The suggested architecture, used a type of anomaly detection to identify Web assaults. Their experimental result gave an accuracy of 97.0%. This work is limited to single-class classification.

[7] proposed a framework for detecting malicious queries on web-applications. The proposed framework comprised both supervised learning and positive and unlabeled learning. The proposed techniques were trained on a payload dataset. The supervised approach was used in learning some of the payload attacks and the Positive and Unbalanced (PU) was used in learning the remaining attacks. The result of their proposed method gave an accuracy of about 93%.

[3] presented a revolutionary approach that uses attention-based Deep Neural Network to precisely detect real-time threats on web-applications. Payload Locating Network (PLN) was employed to recognise the most suspectED locations in massive URL requests and postings while Payload Classification Network (PCN) was utilised to correctly distinguish harmful areas from questionable candidates. This system is limited in detecting just XSS and SQLi attacks.

[1] proposes an attentional RNN framework for detecting payload attacks. The proposed framework develops effective features for detection tasks and provides a visual approach to validate detection outcomes using the attention mechanism. Their simulation results reveal that their suggested model not only achieves high detection rates and low false classification rates but also gives interpretable results.

[8] used hidden Markov algorithms and a bag of words-based (BOW) model in extracting features and more efficiently identifying online threats. The results reveal that, to earlier N-gram extraction feature algorithm studies, BOW has a greater detection rate and a lower false alarm rate at a lower cost. Finally, adequate results in a real-world setting are obtained. The system achieved 96% accuracy. The system is limited to just XSS and SQLi assaults.

Based on benign samples, a supervised detection technique was provided. The Seq2Seq method was chosen and used to detect fraudulent web requests. Meanwhile, the attention approach was introduced to identify the assault payload and emphasize aberrant characters that are labeled. Experimental findings demonstrate that, under the assumption of training a benign sample, the accuracy of their model is 97.02% and the recall is 97.60% [9].

[10] examined the use of ML approaches to exploit Web Application Firewalls (WAFs), and a method for recognizing and blocking assaults. A supplementary machine learning model approach that is dependent on one-class classification and n-gram analysis was utilized to improve the detection and accuracy capabilities of Mod-Security, an open-source and widely used WAF.

To identify web-application payloads in [11], a REGEXNET prototype was built and coupled with HA-Proxy and Node.js. The assessment findings suggest that REGEXNET is successful at restoring web service performance after a zero-day attack. Re-DoS attacks, responsive in responding to assaults in under a minute, and resilient to various Re-DoS attack types, including adaptive ones that are designed to intentionally evade REGEXNET.

[12] suggest a new mapping approach for reducing the characteristics of the dataset utilized for payload attacks. This was done to increase the performance of an existing model. They also attempted to accelerate the training process without negatively impacting detection performance. According to the testing results, training time may be lowered by an average of 24.75 percent. The system gave an accuracy of about 95.5%.

[13] provided a full high interaction web deception system that is aided by a hybrid attack detection module that consists of a deep learning-based classifier paired with a cookie analysis engine that aids in attacker profiling. The primary feature of attacker profiling enabled their system to cope with attackers carrying zero-day payloads while also offering fast session management and scenario-based emulation. Because of its profiling feature, the suggested attack detection module has accuracy of 99.94 percent and takes less time than previous research efforts

METHODOLOGY

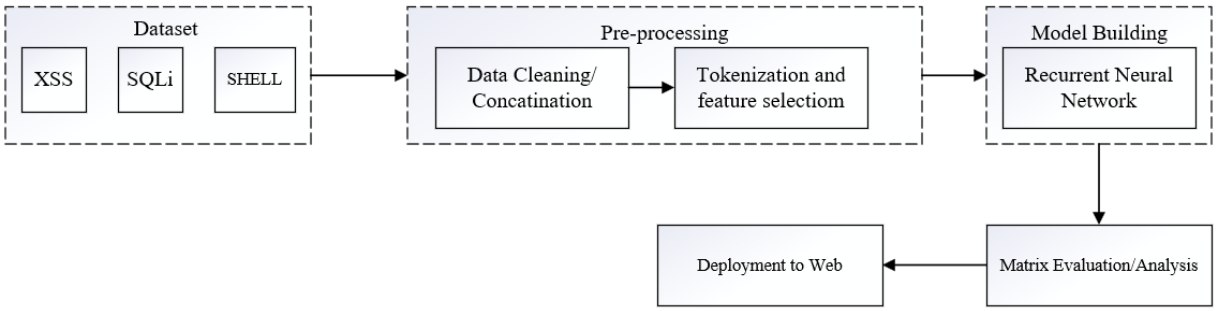

Figure 1: Architectural design

Dataset: The dataset utilized in this paper consists of several web-application assaults. The majority of these attacks are cross-site scripting (XSS), SQL injection (SQLi), and shell assaults. The collection contains a total of one hundred thousand (101,840) scripts, both malicious and non-malicious.

Pre-processing: The pre-processing phase is divided into two stages. The first stage is the data cleaning stage. The data cleaning has to do with the removal of null values, and removal of noise (i.e., parenthesis/non-alphanumeric values). The cleaning of the data was carried out on the malicious scripts. Data concatenation has to do with adding together all the sub datasets to be a single dataset.

The second stage of the pre-processing is that of tokenization and feature selection. Tokenization has to do with the broken down of words (malicious scripts) into tokens (smaller words). Feature selection has to do with the selecting of the most important features in the dataset. We applied feature selection on the tokenized data.

Model Building: The RNN architecture was built using a max word of 5000, sequence length of 250, dropout=0.2. The dropout is a method for reducing overfitting while training the RNN model. Soft max was utilized as the activation function and Binary_Categorical was used as loss function. The RNN model was trained on dense layer of 100 (100 neurons), rooth. The model was trained with a total of 544, 804 trainable parameters. Figure 2 shows the trainable parameters of the RNN model.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 250, 10)           500000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 250, 10)           0
_________________________________________________________________
lstm (LSTM)                  (None, 100)               44400
_________________________________________________________________
dense (Dense)                (None, 4)                 404
=================================================================
Total params: 544,804
Trainable params: 544,804
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: RNN model summary

Algorithm of the proposed RNN model

Step1: Compute the web-based payload dataset as input

step2: compute the current state (ht) of the RNN model by utilizing a set of current input and the initial state.

Step3: The current state (ht) is now set to ht-1.

Step4: Repeat the entire process until a better result is achieved.

Step5: On completion of the iterative steps, the final current state is used in calculating the output.

Step6: The calculated output is then compared to the original output. Therefore, generating the target output and the error.

Step7: The generated error is then backpropagated to the RNN architecture to update it's weights

Matrix Evaluation: The matrix evaluation is used in evaluating the performance of the RNN architecture, in terms of precision, recall, and F1-Score In order to provide a full explanation of the model's performance. This was used to explain the model's accurate and incorrect predictions on the test data. True positive, true negative, false positive, and false negative are all examples of correct and wrong predictions. The matrix evaluation is shown in Table 1 in terms of accuracy, precision, error, recall, and F-measure.

Table 1: Evaluation Matrix

Definition	**Formula**
Accuracy	$\frac{TP+TN}{TP+FN+TN+FP}$
Error	1-accuracy
Recall	$\frac{TP}{TP+FN}$
Precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
F-measure	$\frac{2\,Precison\times Recall}{Precision+Recall}$

RESULT AND DISCUSSION

Jupyter Notebook technology was used in carrying out a simulation in building a robust model for detecting and preventing payload attacks on web applications. The payload dataset was used as input. The payload dataset comprises thousands of scripts/codes written for the purpose of accessing to some vital information or performing some kind of distortion. The thousands of scripts in the payload dataset are of four different categories. These categories are XSS attacks, SQLi, SHELL, and Legitimate scripts. The payload dataset was highly imbalanced, which could have resulted to some mis-classification if the imbalanced problem has not been solved. In other to solve the problem of the data imbalanced, RandomOverSampling technique was applied on the payload dataset. This technique was used in making on the scripts/queries in the dataset will be of the same size. Figure 3 and 4 shows a bar chart of the imbalanced payload dataset and the balanced payload dataset respectively. After solving the problem of data imbalance, some pre-processing is performed on the dataset. The pre-processing has to do with data cleaning and tokenization. We perform data cleaning by removing noise, parenthesis, removal of null values and non-alphanumeric values, this was done so as to have a better trainable data that will be fed into the proposed RNN model. Tokenization was performed on the dataset by splitting the words into

smaller tokens and also transforming the words or scripts into an array of numbers. The transformed data to array can be seen in figure 5. After the transformation of the payload dataset to array, we then feed the result to our proposed recurrent Neural Network Architecture, using total of 100 neurons in the dense_layer, dropout=0.2, betch_size=128, and epoch (number of training steps) to equal 2. The proposed RNN model was built in a total of four layers, one input layer, two hidden layer and an output layer that displays the type of web-based payload attacks. The training process of our proposed model can be seen in figure 7. After training of our proposed RNN model for detecting payloads attacks on web application, we evaluated the performance of our model in terms of accuracy and loss for both training and test data. The evaluation of the model's performance was done using graphs. This can be seen in figure 8 and 9. The result of the classification report can be seen in figure 10.

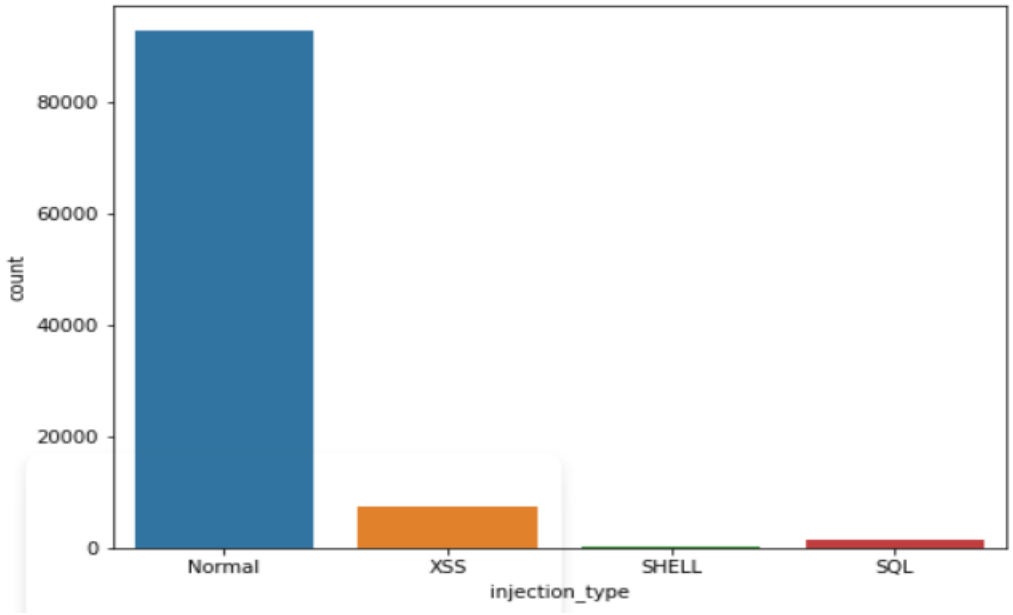

Figure 3: Bar Chart of the Imbalanced dataset

The histogram distribution shows a countplot of the total number of queries in the payload dataset. The countplot shows that the normal scripts/codes that is not harmful to web applications is over 80,000 while the scripts/codes that are of harmful to web applications are below 20,000.

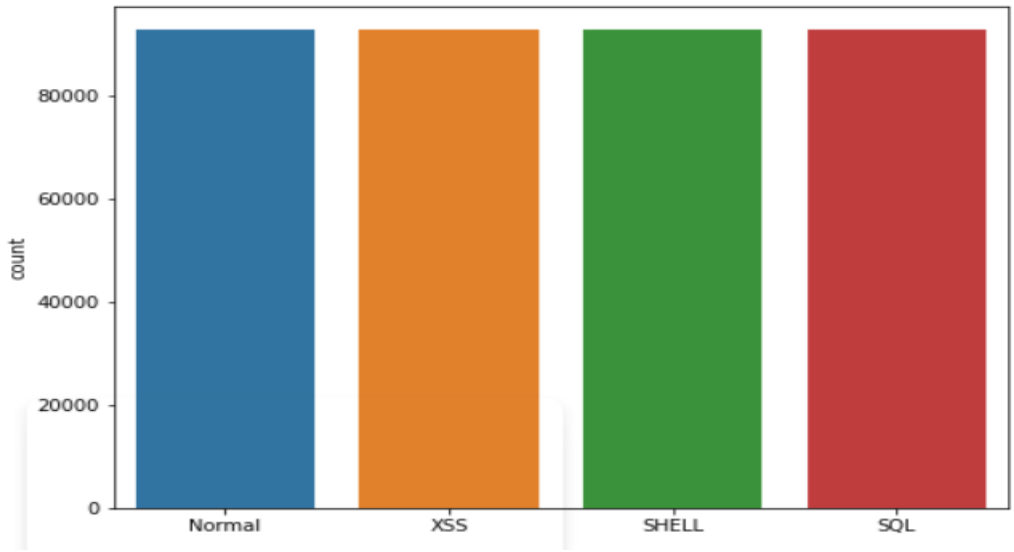

Figure 4: Bar Chart of the balanced dataset

The histogram distribution shows a countplot of the total number of queries in the payload dataset. The countplot shows that the normal scripts/codes that is not harmful to web applications is over 80,000 while the scripts/codes that are of harmful to web applications are also 80,000.

```
array([[    0,     0,     0, ...,     0,     0, 13807],
       [    0,     0,     0, ...,     0,     0, 13808],
       [    0,     0,     0, ...,   485,    58,    22],
       ...,
       [    0,     0,     0, ...,     0,     0,     0],
       [    0,     0,     0, ...,     0,     0,     0],
       [    0,     0,     0, ...,   221,   633,     5]])
```

Figure 5: Transformed Script

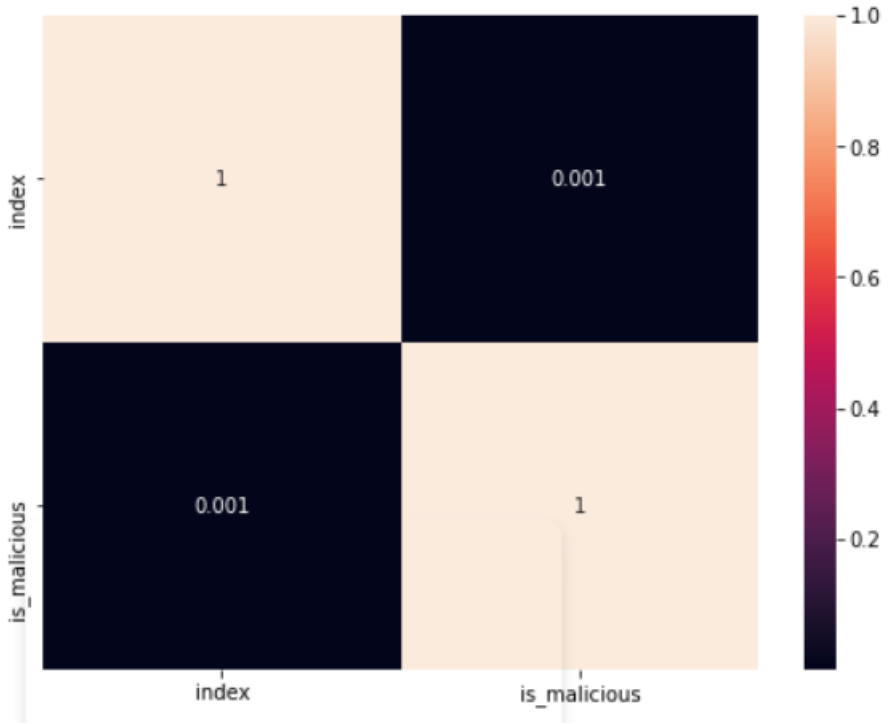

Figure 6: Correlation matrix of the payload dataset.

```
Epoch 1/2
2217/2217 [==============================] - 2100s 946ms/step - loss: 0.1491 - accuracy: 0.9520 - val_loss: 0.0376 - val_accura
cy: 0.9928
Epoch 2/2
2217/2217 [==============================] - 1992s 898ms/step - loss: 0.0448 - accuracy: 0.9900 - val_loss: 0.0316 - val_accura
cy: 0.9937
```

Figure 7: Training process of our proposed model.

Here, the proposed RNN model was trained on two training steps. Each if the training steps shows the accuracy, loss gotten by the model for both training data and testing data.

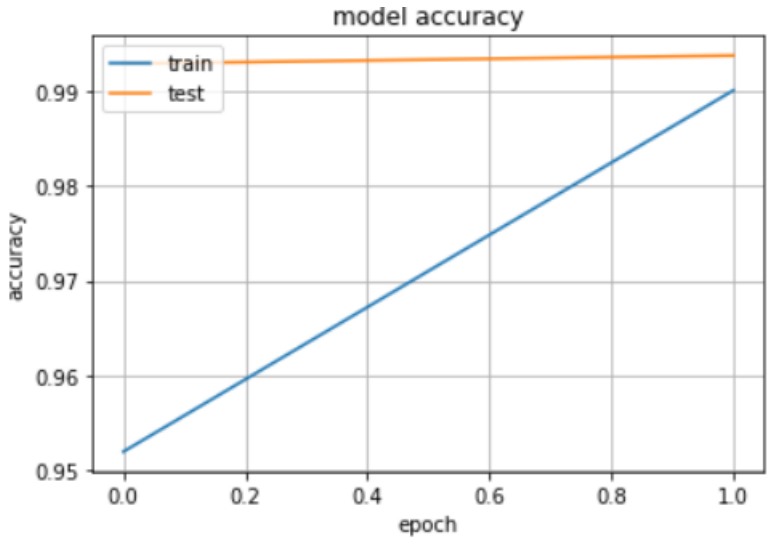

Figure 8: Model Accuracy

Figure 8 shows the accuracy obtained by the model for both training and testing data. The model obtained 99.1% for training data and 99.96% for testing data.

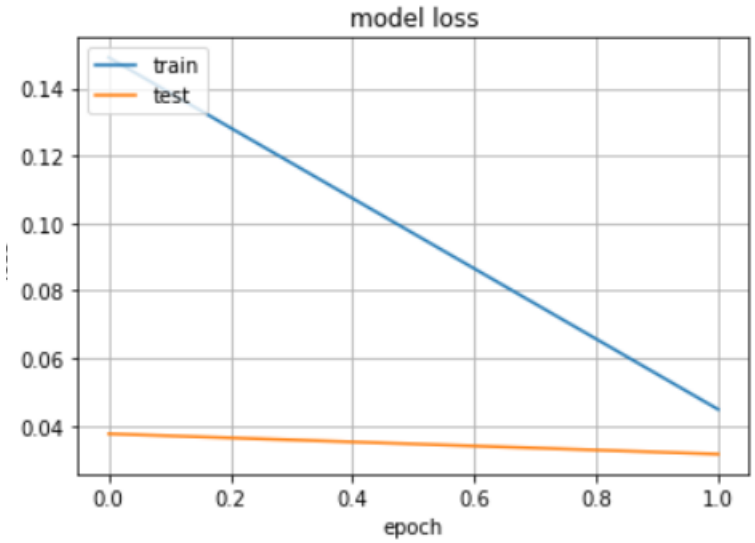

Figure 9: Model Loss

Figure 9 shows the loss obtained by the model for both training and testing data. The model obtained 0.042.% for training data and 0.02% for testing data.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	13901
1	0.82	0.57	0.68	40
2	0.97	0.77	0.86	244
3	0.95	0.94	0.94	1091
accuracy			0.99	15276
macro avg	0.93	0.82	0.87	15276
weighted avg	0.99	0.99	0.99	15276

Figure 10: Classification Report

Figure 11: Security Alert: XSS Detected

CONCLUSION AND FUTURE WORK

Web apps have become a significant platform on which billions of people operate. It has been used for a variety of reasons in accessing numerous platforms over the internet. Because of the rising usage of online apps, cybercriminals have made them a key target, carrying out numerous attacks on web programs such as Cross-site Scripting (XSS), SQLi, and Shell assaults. Because of the prevalence of web application assaults, this study proposes an effective methodology for

identifying and blocking multiple payload web application attacks. In this paper, we trained an RNN model on a dataset including hundreds of distinct web application assaults. This attack makes use of XSS, SQLi, and Shell. The dataset is high unbalanced; thus, we used the RandomOverSampling approach to fix the problem. After resolving the unbalanced problem, we pre-processed the dataset by cleaning it and tokenizing it. We next turned the tokenized data into an array, which we sent as input to our RNN model. Our proposed model was trained on two (2) epochs, where each of the epochs shows the accuracy and loss values obtained by the model for both training and testing data. After training, our proposed RNN model was tested 13901 times, the result of the test gave us a precision 99% for non-payloads attacks, 82% for cross-site scripting, 97% for shell attacks, and 95% for SQLi attacks. We also used a Python flask to construct a strong system for identifying and avoiding various payload assaults on web apps when we released our RNN model to the web. This study is only concerned with web-application assaults. This paper can further be extended by integrating a robust model for detecting payload attacks on android applications. It can also be extended by increasing the number of training epochs from 2 training steps to 10 training steps.

REFERENCES

[1]. Z. Qin, M. Xing-Kong, W. Yong-Jun, "Attentional Payload Anomaly Detector for Web Applications", In *International Conference on Neural Information Processing* (pp. 588-599). Springer, Cham. 2018.

[2]. E. Pantano and C.-V. Priporas, "The effect of mobile retailing on consumers" purchasing experiences: A dynamic perspective," Computers in Human Behavior, vol. 61, pp. 548–555, 2016.

[3]. T. Liu, Y. Qi, L. Shi, J. Yan, "Locate-Then-Detect: Real-time Web Attack Detection via Attention-based Deep Neural Networks", Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), 4725-4731, 2019.

[4]. T. G. Booth and K. Andersson, "Elimination of DoS UDP reflection amplification bandwidth attacks, protecting TCP services," in International Conference on Future Network Systems and Security. Springer, 2015, pp. 1–15.

[5]. Y. Cherdantseva, P. Burnap, A. Blyth, P. Eden, K. Jones, H. Soulsby, and K. Stoddart, "A review of cybersecurity risk assessment methods for SCADA systems," Computers & Security, vol. 56, pp. 1–27, 2016.

[6]. T. Adem. "A novel architecture for web-based attack detection using convolutional neural network." *Computers & Security* 100 (2021): 102096.

[7]. Y. Luo, S. Cheng, C. Liu, F. Jiang, "PU Learning in Payload-based Web Anomaly Detection", Third International Conference on Security of Smart Cities, Industrial Control System, and Communication, 1-5, 2018.

[8]. X. Ren, Y. Hu, W. Kuang,, M. Souleymanou, "A web attack detection technology based on bag of words and hidden Markov model", In *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)* pp. 526-531 2018.

[9]. F. Shi, P. Zhu, X. Zhou, B. Yuan, Y. Fang, "Network attack detection and visual payload labeling technology based on Seq2Seq architecture with attention mechanism", International Journal of Distributed Network Sensors, 6(4), 1-11, 2020.

[10]. G. Betarte, E. Giménez, R. Martínez, A. Pardo, "Improving web application firewalls through anomaly detection", In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 779-784). IEEE.

[11]. Z. Bai, K. Wang, H. Zhu, Y. Cao, X. Jin, "Runtime recovery of web applications under zero-day redos attacks", In *2021 IEEE Symposium on Security and Privacy (SP)* pp. 1575-1588, 2021.

[12]. X. Jin, B. Cui, D. Li, Z. Cheng, C. Yin, "*An improved payload-based anomaly detector for web applications", Journal of Network and Computer Applications, 106, 111–116.* doi:10.1016/j.jnca.2018.01.002

[13]. W. Shahid, B. Aslam, H. Abbas, H. Afzal, S, "A deep learning assisted personalized deception system for countering web application attacks." *Journal of Information Security and Applications* 67 (2022): 103169.