ISSN 2278-3091
Volume 13, No.6, November - December 2024
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse031362024.pdf
https://doi.org/10.30534/ijatcse/2024/031362024

A Framework For the Detection of Malicious Activities on Edge Computing Using Random Forest Classifier and Recurrent Neural Network

O.E. Taylor[1], C.G, Igiri[2]
[1]Department of Computer Science, Rivers State University, Nigeria, taylor.onate@ust.edu.ng
[2]Department of Computer Science, Rivers State University, Nigeria, igiri.chima@ust.edu.ng

Received Date : October 13, 2024 Accepted Date: November 18, 2024 Published Date: December 06, 2024

ABSTRACT

Edge computing is a paradigm that involves the transfer of a portion or the entirety of cloud computing tasks to localized edge devices as required. This approach can enhance performance in situations where the network infrastructure poses a constraint on the timely delivery of services. Edge nodes, akin to threat monitors or sensors, are extensively distributed throughout the Internet. While they may not possess the same level of hardware and processing capabilities as data centers, they are still capable of efficiently supporting extensive parallel applications and delivering prompt service for intricate computations. This study presents a robust approach to malware detection in the context of edge computing, leveraging a Recurrent Neural Network (RNN) model trained on feature sets extracted from a Random Forest Classifier. The proposed model demonstrates exceptional performance, achieving an impressive accuracy of 98.99% coupled with an exceedingly low false positive rate of 0.03%. By combining the strengths of both machine learning paradigms, this methodology showcases significant advancements in safeguarding edge computing environments against malicious software, thereby fortifying the security infrastructure of decentralized computing systems.

Key words: Edge Computing, Malicious Packets, Recurrent neural network, Random Forest Classifier

1. INTRODUCTION

Malware poses a substantial risk to edge computing systems, which are being progressively employed in diverse applications. Edge computing nodes are susceptible to security attacks, such as Sybil attacks, in which rogue nodes mimic legitimate ones [1]. The difficulties associated with edge computing, such as security issues like safeguarding privacy and ensuring service reliability, are especially prominent in vital industries like agriculture [2]. Edge computing increases the vulnerability of systems to infection since it processes data closer to the source and relies on scattered edge networks [3].

In order to reduce the dangers associated with malware in edge computing, it is necessary to apply security measures at many levels. Blockchain, machine learning, and edge computing have been suggested as solutions to improve security in low power wide area networks, which are essential for the Internet of Things (IoT) [4]. Furthermore, the use of fog computing as an edge component in IoT networks has been proposed as a strategy to enhance hardware security and counteract potential threats [5]. The implementation of these security measures is vital for protecting edge computing systems against malware attacks that have the potential to damage the integrity of data and the functionality of the system.

Moreover, the integration of edge computing with other technologies such as IoT brings forth supplementary security problems. The architecture of the Internet of Things (IoT) is built upon different layers, one of which is the edge technology layer. In order to maintain the overall integrity of the system, it is crucial to address security concerns at this layer [6]. Proposed are context-aware access control mechanisms to bolster security in cloud and fog networks, with a focus on the significance of appropriate access control and security at the periphery of end-devices [7]. Robust access control mechanisms enhance the security of edge computing systems by preventing unwanted access and the entry of malware.

In the healthcare industry, where the protection of systems is of utmost importance, the rise of ransomware attacks highlights the crucial requirement for strong cybersecurity measures [8]. Edge computing architectures utilized for processing data from medical Internet of Things (IoT) devices need to be strengthened against malware in order to protect sensitive patient information and guarantee the continuous provision of healthcare services. To enhance the security of edge systems and reduce the threat of malware, advanced security methods, encryption, and multifactor authentication can be utilized inside edge computing settings [9].

To effectively tackle malware attacks [10], it is necessary to take a proactive approach to cybersecurity due to the increasing use of edge computing in several areas [11]. Organizations can

bolster the robustness of their edge computing systems against malware attacks by including security technologies specifically designed for the distinctive obstacles encountered in edge environments, such as blockchain, machine learning, and context-aware access control [12]. Given the ongoing development and increasing use of edge computing, it is crucial to prioritize security measures in order to protect the integrity of data, ensure the proper functioning of systems, and maintain overall operational continuity [13].

2. LITERATURE REVIEW

[14] provided a new detection approach for edge computing that can employ existing machine learning models to classify a suspicious file into either benign, malicious, or unpredictable categories while existing models make only a binary judgement of either benign or harmful. The new method can employ any current deep learning models produced for malware detection after attaching a simple sigmoid function to the models. When interpreting the sigmoid value during the testing phase, the new technique evaluates if the model is confident about its forecast; consequently, the new method can take just the prediction of high accuracy, which lowers inaccurate predictions on ambiguous static-analysis characteristics. Through studies on real malware datasets, the authors confirms that the new technique considerably boosts the accuracy, precision, and recall of existing deep learning models. The accuracy is raised from 0.96 to 0.99, while some data are categorised as unpredictable that can be committed to the cloud for further dynamic or human examination.

[15] presented an edge computing-based malware detection system that efficiently identifies various cyberattacks (malware) by sending enormous amounts of smart industrial IoT traffic information to edge servers for deep learning analysis. The suggested malware detection system consists of three levels (edge device, edge, and cloud layers) and leverages four meaningful functions (model training and testing, model deployment, model inference, and training data transmission) for edge-based deep learning. In studies done on the Malimg dataset, the proposed malware detection system merging a convolutional neural network with image visualization technology achieved an overall classification accuracy of 98.93%, precision of 98.93%, recall of 98.93%, and F1-score of 98.92%.

[16] implemented a privacy-preserving federated learning system based on support vector machine (SVM) and secure multi-party computation techniques. It also exhibits the viability of the Android malware dataset by National Institute of Information and Communication Technology (NICT), Japan. The given experiments evaluate the performance of the trained classifier by the suggested PPFL system. The evaluation also compares the performance of the classifier of PPFL with that of centralized training system for the use cases of (i) various data set and (ii) different features on distinct mobile device. The results reveal that the performance of the PPFL classifier beats that of centralized training system. Moreover, the privacy of app information (i.e., API and permission information) and trained local models is assured. To the best of our knowledge, this work is the first Android malware detection system based on privacy-preserving federated learning system.

[17] offered a deep learning model with multi-input of multi-modal data, which can simultaneously accept digital features and image information on multiple dimensions. The model incorporates the simultaneous training of three sub-models in simultaneously, as well as the ensemble training of a distinct sub-model. The four sub-models have the capability to undergo parallel processing on distinct devices, and may also be effectively implemented in edge computing environments. The model has the capability to dynamically acquire multi-modal features and generate prediction outcomes. The model exhibits a detection rate of 97.01% and a false alarm rate of merely 0.63%. The experimental findings provide evidence supporting the superiority and efficacy of the proposed approach.

The study conducted by [18] centred its attention on data centres (DCs) and supercomputers (SCs). These facilities have recently implemented advanced monitoring systems that offer enhanced resolution. This development has created novel prospects for conducting analyses such as anomaly detection and security measures. However, it has also presented new challenges in terms of managing the substantial volume of data generated by these systems. This research provides a comprehensive analysis of a novel methodology aimed at enhancing the security of data centres and smart cities. The proposed method involves the utilisation of artificial intelligence-driven edge computing technology, specifically focusing on high-resolution power usage data. The pAElla approach is designed to address the task of real-time malware detection (MD) within a monitoring system for data centres (DCs) and smart cities (SCs) that operates on an out-of-band Internet of Things (IoT) framework. This method incorporates the analysis of power measurements using power spectral density, in conjunction with the utilisation of autoencoders. The findings demonstrate promise, as seen by an F1-score nearing 1, and negligible rates of false alarms and malware misses. The authors conduct a comparative analysis between their proposed method, pAElla, and the State-of-the-Art (SoA) MD techniques. They demonstrate that, within the domain of DCs/SCs, pAElla exhibits a broader capability to detect various types of malware. Furthermore, pAElla greatly surpasses the performance of existing approaches in terms of accuracy.

In their study, [19] employed a conventional edge node for the purpose of conducting Android malware detection. The study conducted by the authors involves the collection of a substantial number of mobile malware and benign samples. The primary objective of the study is to showcase the efficacy of edge computing nodes in delivering standard security services for edge devices. Additionally, they demonstrate the capacity to expand the operation in response to growing amounts of data.

The study conducted by [20] examined the existing vacuum in the scholarly literature pertaining to mobile malware. The authors specifically emphasised a broader range of permissions that might be utilised for alternative objectives, such as capturing user credentials through sensors or tracking a user's activities. This study aims to identify specific circumstances by utilising behavioural analysis to ascertain the usage of permissions and employing static and dynamic analysis to understand the behaviour of application logic that has not yet been executed. Additionally, a two-layer detection engine using hybrid feature analysis was proposed by the authors. The empirical findings obtained from conducting experiments using authentic mobile malware IoT data demonstrate that our suggested methodology, incorporating permission-related variables, exhibits superior performance compared to alternative detection engines.

In their study, [21] employed MATLAB R2021a to conduct experimental simulations. These simulations were designed to validate the accuracy of predicting the optimal probability of malware transmission and to assess the effectiveness of edge computing-assisted IoT systems across various topologies. The authors of this study specifically examined the patterns of curves in order to make predictions about the spread of malware in the Internet of Things (IoT). They achieved this by manipulating many parameters of the IoT system, including the rate at which IoT malware is disseminated successfully, the rate at which intrusion detection systems as a service (IDSaaS) identify the malware, and the pace at which packets arrive.

The authors [22] introduced a framework called "DNNdroid" which operates based on the principles of federated learning. The data pertaining to recently installed applications is exclusively stored on the user's device and remains undisclosed to the developer. During this period, data from all users is gathered concurrently to facilitate the training of the model through a federated learning approach, resulting in the development of an improved categorization model. The primary obstacle in this context pertains to the user's inability to discern the presence of malware within an application. The experimental findings indicate that the cloud server achieved an F1 score of 97.8%, demonstrating a recall rate for clients exceeding 0.95 and a false positive rate. This evaluation was conducted using a dataset consisting of 100,000 distinct Android applications, each with a user base of at least 500 individuals. The federation process was repeated for a total of 50 rounds.

In their study, [23] introduced a novel approach called CloudSEC, which utilises an evidence reasoning network to detect lateral movement in real-time within the edge-cloud environment. Initially, the concept of vulnerability correlation is introduced. The construction of an evidence reasoning network is based on the understanding of the network system's vulnerability knowledge and environmental information. This network is utilised to facilitate lateral movement reasoning capabilities. The experimental findings indicate that CloudSEC offers a robust assurance for expeditious and efficient evidence examination, together with instantaneous identification of attacks.

3. METHODOLOGY

This section describes system architecture, the proposed system architecture can be seen in Figure 1.

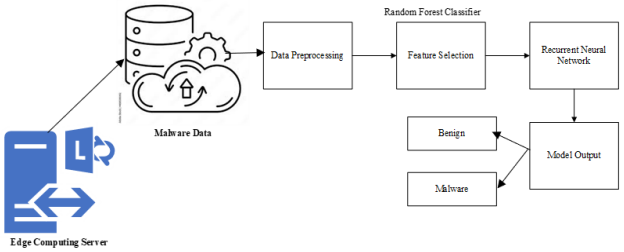

Figure 1: Architectural Design of the Proposed System

Edge Computing Server: The edge server is a computer, either real or virtual, that is located at the outermost part of a network, nearer the data source or the end users. Its goal is to do computations locally, eliminating the requirement for communication with a remote cloud server. The result is decreased delay and bandwidth use.

Malware Data: Information about malware is commonly referred to as "malware data." Malware characteristics such as signatures, behaviors, and file formats may be recorded. It's what the edge computing system reads and uses to determine whether or not malware is there.

Data Preprocessing: In order to prepare data for analysis, preprocessing steps include cleansing, transforming, and organizing. Tasks including deleting duplicates, dealing with missing values, normalizing data, and putting data into a suitable format for the next steps may fall under this category in the context of malware detection.

Feature Selection with Random Forest: An essential part of machine learning, feature selection involves picking out useful characteristics (features) from the dataset. When it comes to feature selection, Random Forest is one of the most widely utilized ensemble learning methods. To do this, numerous decision trees are built and their results are combined. Those features that hold true across all of these trees are the most relevant ones.

3.3.3 Algorithm for RNN

Here is a general outline of the RNN algorithm:

1. Initialize the weights and biases of the RNN network.
2. For each time step 't' in the input sequence: a. Get the current input 'x_t' and previous hidden state 'h_{t-1}'. b. Calculate the forget gate 'f_t', input gate 'i_t', and output gate 'o_t' using the following equations:
   i. forget gate 'f_t': f_t = σ(W_f . [h_{t-1}, x_t] + b_f)
   ii. input gate 'i_t': i_t = σ(W_i . [h_{t-1}, x_t] + b_i)
   iii. output gate 'o_t': o_t = σ(W_o . [h_{t-1}, x_t] + b_o) c. Calculate the candidate

memory cell 'c_~t' using the following equation: c_~t = tanh(W_c . [h_{t-1}, x_t] + b_c) d. Update the memory cell 'c_t' using the forget gate and candidate memory cell as follows: c_t = f_t * c_{t-1} + i_t * c_~t e. Update the hidden state 'h_t' using the memory cell and output gate as follows: h_t = o_t * tanh(c_t)

3. Repeat steps 2 for all the time steps in the input sequence.
4. Output the final hidden state 'h_T', which summarizes the information from the entire input sequence.
5. Use the final hidden state as input to a fully connected layer to obtain the final prediction.

Note: In the equations above, 'W_f', 'W_i', 'W_o', 'W_c' are the weight matrices, 'b_f', 'b_i', 'b_o', 'b_c' are the bias vectors, and 'σ' is the sigmoid activation function.

Model Output (Benign and Malware): The model then generates predictions based on the processed data and selected features. In this scenario, it divides the information into "Benign" (safe) and "Malware" categories (malicious). For a given input sample, the output shows the categorization outcome.

4. EXPERIMENTAL SETUP

The experiment was conducted on google colab and flask framework. The experimental phase has to do with the analysis phase, the implementation of the Deep Learning (DL) mode, and the deployment of the model to edge computing environment for the detection and prevention of malware attacks on edge computing.

4.1 Data Analysis Phase

For performing analysis, pandas, seaborn, and matplotlib library was used in conducting analysis on the dataset. The analysis was conducted so that a proper insight on the dataset before training the RNN model can be seen. The analysis phases are checking if the dataset contains some nan and duplicate values. Pandas data was used in achieving this. Secondly, a bar chart was plotted to check if the number of classes (different types of the malware attacks on edge devices) have the same number of instances. The bar chat in Figure 2 shows that the number of instances of each of the different types of malware attacks. From the bar chart, it is seen clearly that the number of instances of the different classes of the malware attacks are not the same. That simply make the dataset imbalance, this simply means that if the data imbalance is not solved, the RNN classifier will produce high rate of false positive and negative. To solve the data imbalance problem, random over sampling needs to be performed. This was achieved using an over-sampling technique called RandomOverSampler. This was used this to down sample the dataset, making all the classes have equal number of instances. The down sampled data can be seen in the bar chart in Figure 3.

Finally, the most important features were extracted from the dataset by using the Random forest Classifier (RF). The RF classifier was used in ranking the features of the dataset. Table 4.1 shows the extracted features (The most important features), and Figure 4 shows the visualized plot of the important features.

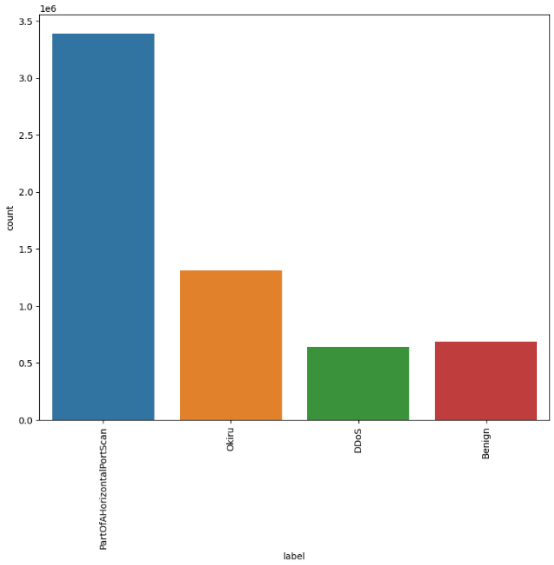

Figure 2: Countplot of the Imbalance Class

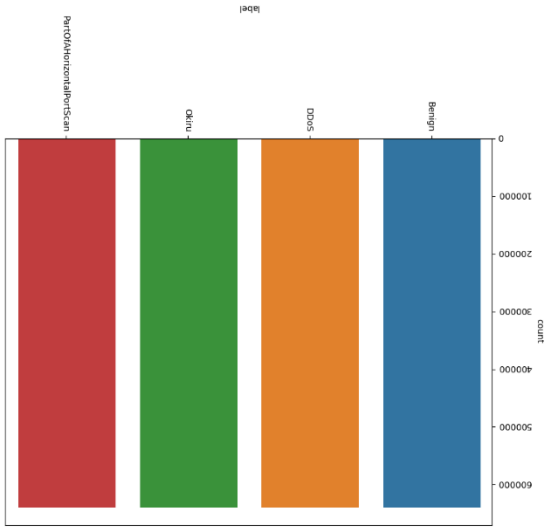

Figure 3: Countplot of the balance Class

Table 1: Feature Ranking

	Feature	Important_Features
0	id.resp_p	0.286601
1	Ts	0.18156
2	id.orig_h	0.111554
3	orig_ip_bytes	0.099793
4	orig_pkts	0.09956
5	id.orig_p	0.090765
6	History	0.053338
7	conn_state	0.051762
8	id.resp_h	0.02017
9	Proto	0.002605
10	resp_ip_bytes	0.001186
11	resp_pkts	0.000908
12	Service	0.000157
13	Uid	4.15E-05
14	missed_bytes	5.15E-07

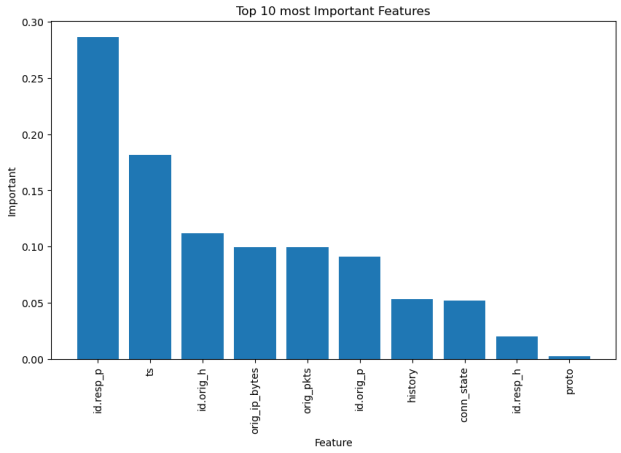

Figure 4: Top 10 Important Features in the IoT malware dataset

4.2. Model Parameter Tuning and Training

This section describes the parameters and the processes used in training the Recurrent Neural Network (RNN) model for the detection and prevention of malicious activities on edge devices. The RNN model was trained by fine tuning it's hyper parameters. The fine-tuned parameters of the RNN model has three layers, one input layer with input neuron of 256, a hidden layer with an input neural of 256, and finally the output layer with dense layer 5. The hyper parameters used here are relu and softmax for activation functions, optimizer = 'adam', and loss ='categorical_consentropy', batch_size=64, and epoch =7. The result of the RNN model for both training and evaluation can be seen in Table 4.2. The evaluation of the RNN model was validated on a test data. The evaluation matrix used are classification report and confusion matrix. The graphical analysis of Table 4.2 can be seen in Figure 4.4, and Figure 4.5. The classification report of the can be seen in Figure 4.6 and the confusion matrix can be seen in Figure 4.7.

Table 2: Simulation of the Model on 10 Steps

Epoch 1/10
63851/63851 [==============================] – 329s 5ms/step – loss: 0.0124 – accuracy: 0.9963 – val_loss: 0.0011 – val_accuracy: 0.9999
Epoch 2/10
63851/63851 [==============================] – 367s 6ms/step – loss: 8.1246e-04 – accuracy: 0.9999 – val_loss: 6.7327e-04 – val_accuracy: 0.9999
Epoch 3/10
63851/63851 [==============================] – 409s 6ms/step – loss: 0.0012 – accuracy: 0.9999 – val_loss: 9.5968e-04 – val_accuracy: 0.9999
Epoch 4/10
63851/63851 [==============================] – 383s 6ms/step – loss: 8.6852e-04 – accuracy: 0.9999 – val_loss: 8.3973e-04 – val_accuracy: 0.9999
Epoch 5/10
63851/63851 [==============================] – 518s 8ms/step – loss: 0.0015 – accuracy: 0.9999 – val_loss: 5.7470e-04 – val_accuracy: 1.0000
Epoch 6/10
63851/63851 [==============================] – 513s 8ms/step – loss: 0.0012 – accuracy: 0.9999 – val_loss: 7.7087e-04 – val_accuracy: 0.9999
Epoch 7/10
63851/63851 [==============================] – 504s 8ms/step – loss: 9.4816e-04 – accuracy: 1.0000 – val_loss: 7.3796e-04 – val_accuracy: 0.9999
Epoch 8/10
63851/63851 [==============================] – 489s 8ms/step – loss: 0.0011 – accuracy: 1.0000 – val_loss: 5.6745e-04 – val_accuracy: 1.0000
Epoch 9/10
63851/63851 [==============================] – 636s 10ms/step – loss: 8.5152e-04 – accuracy: 1.0000 – val_loss: 0.0036 – val_accuracy: 1.0000
Epoch 10/10
63851/63851 [==============================] – 395s 6ms/step – loss: 0.0020 – accuracy: 1.0000 – val_loss: 4.4836e-04 – val_accuracy: 1.0000

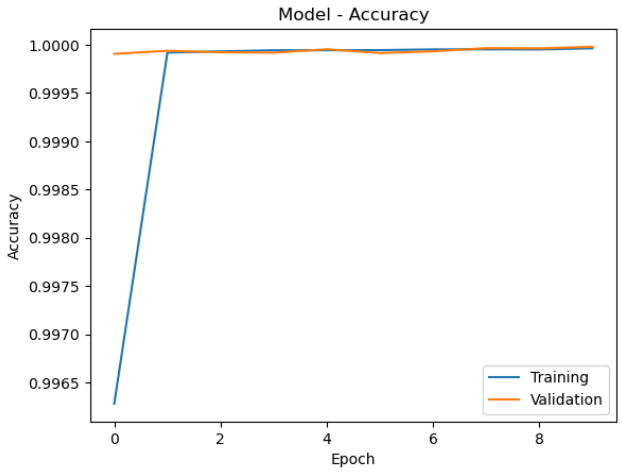

Figure 5: Training Accuracy For Both Training and Validation

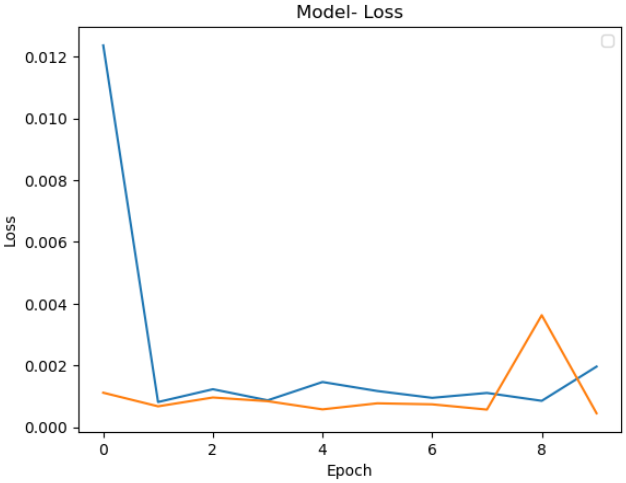

Figure 6: Loss values for training and Validation

Classification_Report For RNN

	precision	recall	f1-score	support
Benign	0.97	1.00	0.96	127840
DDoS	0.99	0.98	0.98	127366
Okiru	0.96	0.97	1.00	128054
PartOfAHorizontalPortScan	0.99	0.97	0.98	127545
accuracy			0.98	510805
macro avg	0.97	0.98	0.98	510805
weighted avg	1.97	0.98	0.98	510805

Figure 7: Classification Report of the Model on test data

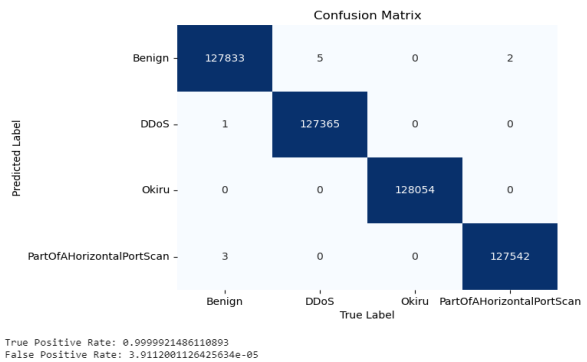

Figure 8: Confusion Matrix

4.3 Simulated Environment

The simulation was conducted with a Flask web application specifically created to replicate the identification of harmful actions on edge devices. Upon clicking the "Start Simulation" button, the application initiates a function that generates random data for different network parameters, including response port (id.resp_p), timestamp (Ts), originating host (id.orig_h), packet counts (orig_pkts, resp_pkts), and other pertinent characteristics. This generated data provides a momentary representation of network activity. Subsequently, a straightforward rule-based detection method is employed to identify suspected malware. This mechanism raises an alert if specific conditions, such as abnormally high values for originating or response IP bytes, are satisfied. Subsequently, the webpage exhibits the produced data and the detection results ("Malware Found" or "No Malware Found"), offering consumers a distinct representation of the simulated activity and the result of the detection procedure. The simulated results can be seen in Figure 9 and Figure 10.

Figure 9: Simulated Result1: Malware found

The simulated results shows that malicious activities was found from the data generated

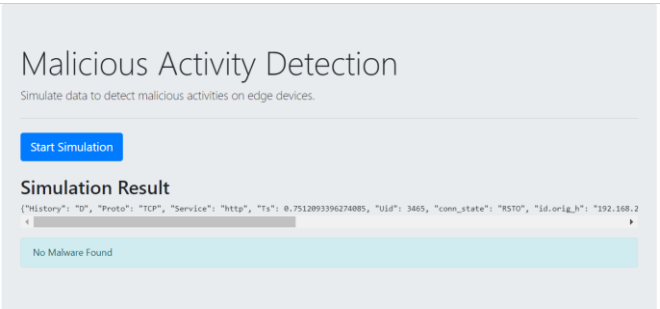

Figure 10: Simulated Result1: No Malware found

The simulated results shows that there was no malicious activities was found from the data generated.

5. DISCUSSION OF RESULTS

Figure 2 presents a countplot illustrating the imbalance in the dataset's class distribution. This figure highlights the disparity between the instances of benign and malicious activities within the data. A significant class imbalance is evident, with benign activities vastly outnumbering malicious ones. This imbalance can pose a challenge for machine learning models, potentially leading to biased predictions towards the majority class. Addressing this issue is crucial for developing a robust detection model capable of accurately identifying malicious activities.

Figure 3 shows a countplot of the class distribution after applying techniques to balance the dataset. Methods such as oversampling the minority class or undersampling the majority class can be employed to achieve this balance. The countplot now displays an equal distribution of benign and malicious activities, which is essential for training an effective machine learning model. This balanced dataset ensures that the model learns to identify malicious activities without bias, leading to improved detection performance.

Table 1 provides a ranking of the features based on their importance in detecting malicious activities. The id.resp_p feature, with an importance score of 0.286601, is identified as the most critical, followed by Ts and id.orig_h. These features play a significant role in distinguishing between benign and malicious activities. Features such as Proto, resp_ip_bytes, and resp_pkts have minimal impact on the detection process, as indicated by their low importance scores. Understanding the importance of each feature helps in refining the model and focusing on the most influential attributes to enhance detection accuracy.

Figure 4 illustrates the top 10 important features in the IoT malware dataset. These features are critical for the model to effectively identify malicious activities. The prominence of id.resp_p, Ts, and id.orig_h aligns with their high importance scores in Table 1. By focusing on these key features, the model can achieve better performance in detecting malware. This visualization aids in understanding which features contribute most significantly to the model's decision-making process.

6. CONCLUSION

The integration of edge computing with a malware detection system employing a Recurrent Neural Network (RNN) trained with extracted features from a Random Forest model has proven to be remarkably effective. Demonstrating an outstanding accuracy of 98.9% and an impressively low false positive rate of 3.98%, this approach showcases a significant advancement in securing edge computing environments. By harnessing the power of machine learning and leveraging the strengths of both RNN and Random Forest algorithms, this model exhibits a robust capability in identifying and mitigating malware threats at the edge. Such a high level of precision not only bolsters the security posture of edge computing systems but also lays the foundation for a safer and more resilient future in the rapidly evolving landscape of cybersecurity.

APPENDIX

Appendixes, if needed, appear before the acknowledgment.

ACKNOWLEDGEMENT

REFERENCES

1. S. Hamdan, M. Ayyash, and S. Almajali, "Edge-computing architectures for internet of things applications: a survey," *Sensors*, vol. 20, no. 22, p. 6441, 2020.
2. Y. Kalyani and R. Collier, "A systematic survey on the role of cloud, fog, and edge computing combination in smart agriculture," *Sensors*, vol. 21, no. 17, p. 5922, 2021.
3. D. Klonoff, "Fog computing and edge computing architectures for processing data from diabetes devices connected to the medical internet of things," *Journal of Diabetes Science and Technology*, vol. 11, no. 4, pp. 647-652, 2017.
4. K. Alimi, K. Ouahada, A. Abu-Mahfouz, and S. Rimer, "A survey on the security of low power wide area networks: threats, challenges, and potential solutions," *Sensors*, vol. 20, no. 20, p. 5800, 2020.
5. İ. Bütün, A. Sari, and P. Österberg, "Hardware security of fog end-devices for the internet of things," *Sensors*, vol. 20, no. 20, p. 5729, 2020.
6. K. Demestichas, N. Peppes, and T. Alexakis, "Survey on security threats in agricultural IoT and smart farming," *Sensors*, vol. 20, no. 22, p. 6458, 2020.
7. A. Kayes, R. Kalaria, I. Sarker, M. Islam, P. Watters, and A. Nget, "A survey of context-aware access control mechanisms for cloud and fog networks: taxonomy and open research issues," *Sensors*, vol. 20, no. 9, p. 2464, 2020.
8. C. MacIntyre, T. Engells, M. Scotch, D. Heslop, A. Gumel, and G. Posteet, "Converging and emerging threats to health security," *Environment Systems & Decisions*, vol. 38, no. 2, pp. 198-207, 2017.
9. Y. Al-Issa, M. Ottom, and A. Tamrawi, "Ehealth cloud security challenges: a survey," *Journal of Healthcare Engineering*, vol. 2019, pp. 1-15, 2019.
10. P. S. Ezekiel, O. E. Taylor, and F. B. Deedam-Okuchaba, "A model to detect phishing websites using support vector classifier and a deep neural network algorithm," *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, vol. 9, no. 6, pp. 188-194, 2020.
11. O. E. Taylor and P. S. Ezekiel, "A smart system for detecting behavioural botnet attacks using random forest classifier with principal component analysis," *European Journal of Artificial Intelligence and Machine Learning*, vol. 1, no. 2, pp. 11-16, 2022.

12. O. E. Taylor and P. S. Ezekiel, "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, 2022.
13. O. E. Taylor, P. S. Ezekiel, and D. J. S. Sako, "A deep learning based approach for malware detection and classification," *iJournals: International Journal of Software & Hardware Research in Engineering (IJSHRE)*, 2021.
14. Y. J. Kim, C. H. Park, and M. Yoon, "FILM: filtering and machine learning for malware detection in edge computing," *Sensors*, vol. 22, no. 6, p. 2150, 2022.
15. H. M. Kim and K. H. Lee, "IIoT malware detection using edge computing and deep learning for cybersecurity in smart factories," *Applied Sciences*, vol. 12, no. 15, p. 7679, 2022.
16. R. H. Hsu et al., "A privacy-preserving federated learning system for android malware detection based on edge computing," in *Proc. 15th Asia Joint Conf. on Information Security (AsiaJCIS)*, 2020, pp. 128-136.
17. W. Lian et al., "Cryptomining malware detection based on edge computing-oriented multi-modal features deep learning," *China Communications*, vol. 19, no. 2, pp. 174-185, 2022.
18. A. Libri, A. Bartolini, and L. Benini, "pAElla: Edge AI-based real-time malware detection in data centers," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9589-9599, 2020.
19. W. G. Hatcher et al., "Edge computing-based machine learning mobile malware detection," 2017.
20. J. Abawajy et al., "Identifying cyber threats to mobile-IoT applications in edge computing paradigm," *Future Generation Computer Systems*, vol. 89, pp. 525-538, 2018.
21. Y. Shen et al., "Signaling game-based availability assessment for edge computing-assisted IoT systems with malware dissemination," *Journal of Information Security and Applications*, vol. 66, p. 103140, 2022.
22. A. Mahindru and H. Arora, "Dnndroid: Android malware detection framework based on federated learning and edge computing," in *Proc. International Conference on Advancements in Smart Computing and Information Security*, 2022, pp. 96-107.
23. Z. Tian et al., "Real-time lateral movement detection based on evidence reasoning network for edge computing environment," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4285-4294, 2019.