Intrusion Detection Systems Based on Logarithmic Autoencoder and XGBoost

Wenfeng Xu and Yongxian Fan

School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China

Correspondence should be addressed to Yongxian Fan; yongxian.fan@gmail.com

Received 4 December 2021; Revised 22 March 2022; Accepted 30 March 2022; Published 11 April 2022

Copyright © 2022 Wenfeng Xu and Yongxian Fan. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

An intrusion detection system (IDS) is a network security device that performs real-time monitoring of network transmissions and sends out alarms or takes active response measures when suspicious transmissions are found. In this regard, many researches have combined traditional machine learning models with other optimization algorithms to improve intrusion detection performance. However, although the existing intrusion detection model can effectively improve the performance of the model, there are still problems such as unsatisfactory detection accuracy and data preprocessing operations that may lead to a decrease in accuracy. To solve this problem, in this paper, we have proposed a novel intrusion detection system model based on logarithmic autoencoder (LogAE) and eXtreme Gradient Boosting (XGBoost). First, we build LogAE to learn the hidden features of the input data to reconstruct new data similar to the training samples, with the purpose of highlighting important features. It is worth mentioning that LogAE is not necessary to normalize the training data. This is because we add a logarithmic layer to learn this mapping. Then, XGBoost is used as a classifier to identify the data set that combines the original data set with the generated data set. In the experiment, our proposed model is evaluated on the UNSW-NB15 data set and CICIDS2017 data set. Additionally, we use accuracy, recall, precision, F1-score, and runtime as evaluation metrics. For detection performance, the detection accuracy of our proposed model is 95.11% for UNSW-NB15 and 99.92% for CICIDS2017, which is better than most state-of-the-art intrusion detection methods. Meantime, the runtime of our proposed model is the lowest for UNSW-NB15.

1. Introduction

In modern society, people are increasingly inseparable from the Internet. With the increasing popularity of the Internet, the problem of network security has become more and more prominent. Relying solely on firewalls cannot effectively defend against network attacks. Therefore, intrusion detection systems as the second line of defense have received more and more attention. Generally speaking, intrusion detection systems can be divided into network intrusion detection (NIDS) [1] and host intrusion detection (HIDS) [2] according to the type of target system. The data source of NIDS is the data packet on the network, which detects all network transmissions in this network segment. The HIDS function is similar to a virus firewall. It runs in the background of the system to be protected and detects host activities.

With the increasing complexity of the network environment, detection systems have gradually shifted from rule- and expert-based methods to machine-learning-based methods, such as decision trees [3], support vector machines [4], XGBoost [5], neural networks [6], and so on. XGBoost, as one of the supervised learning models, still has unique advantages in the scenario of limited training samples, short training time, and lack of parameter tuning knowledge when neural networks, especially deep neural networks, are becoming more and more popular. It has achieved good results on issues such as recommendation, search ranking, user behavior prediction, click-through rate prediction, and intrusion detection systems. Compared with deep neural networks, XGBoost can handle tabular data better, has stronger interpretability, and has the advantages of easy parameter adjustment and input data invariance.
In addition, autoencoder (AE) [7] is playing an important role in many fields. AE is a type of feedforward neural network, which was initially mainly used for data dimensionality reduction and feature extraction. As the technology continues to develop, it is now also used in generative models, which can be used to reconstruct data and so on. At the same time, the AE trained by a specific data distribution can restore the data belonging to the distribution well, while for the data that does not belong to the distribution, its restoration error is relatively large. This characteristic enables AE to be applied in the field of classification.

However, in these algorithms and models, most of them require preprocessing operations such as data normalization before performing feature engineering and training the classifier. Although this is helpful to improve the performance of the model, it may cause loss of data accuracy and detection accuracy for intrusion detection data sets. Data normalization may also not be conducive to the adjustment of the data set, such as adding or deleting data requires redefining the entire training process, resulting in longer runtime.

In this paper, we have proposed a novel intrusion detection model based on LogAE and XGBoost. In the proposed model, we first build LogAE, which added a logarithmic layer before AE to replace the normalized mapping, to learn the hidden features of the input data to generate new data similar to the training sample, so as to make the important features prominent. Then, XGBoost is used to classify the training set that combines the original data set with the new generated data set. Our contributions to this research are as follows:

1. We have proposed an intrusion detection model based on LogAE and XGBoost, in which we use LogAE to generate the reconstructed data based on the original data and use XGBoost to detect whether a sample that combines the reconstructed data and the original data belongs to normal or attack.

2. We have proposed an improved autoencoder, named LogAE, which combines the logarithmic neural and autoencoder. In LogAE, the data normalization operation is completed by the logarithmic layer instead of some normalization operations such as min-max normalization.

3. We have conducted comparative experiments on UNSW-NB15 data sets and the CICIDS2017 data set, illustrating that our proposed model obtains higher detection performance and outperforms most of the state-of-the-art methods in running time.

In this paper, the other sections are as follows. Section 2 proposes a review of related work in the intrusion detection system. Section 3 supplies the details of our proposed detection model. Section 4 presents experimental details and the comparison of our model and other machine learning algorithms of our model. Section 5 indicates the conclusion and an overall review of our research results.

2. Related Work

Numerous researches on intrusion detection based on XGBoost and AE have been conducted. For XGBoost and AE, they can be used not only as classifiers but also for feature selection.

Sumiya Thaseen Ikram et al. [8] proposed an ensemble intrusion detection model including long short-term memory (LSTM), backpropagation network (BPN), and multilayer perceptron (MLP). There were two core modules in this model: learning algorithm module and evaluation module. The learning algorithm module consisted of LSTM, BPN, and MLP, which were used to obtain training data as input. Three model output was sent to the XGBoost in the evaluation module. Jiang Hui and others [9] proposed particle swarm optimization (PSO) XGBoost IDS model to address the false positive issue. In this model, PSO was used to adaptively search for the optimal parameters of XGBoost, and XGBoost was regarded as a classifier. Karthikraja and others [10] mainly focused on the feature of the attack in the IDS. They adopted XGBoost as a feature selection model to choose important features of the attacks in the data set and a bidirectional long short-term model (Bi-LSTM) as a classifier to identify attacks. Chunhe Song and others [11] proposed an intrusion detection system method based on the combination of the deep learning method and feature-based method. This method uses a Bayesian algorithm to optimize the superparameters of XGBoost to avoid the performance degradation of XGBoost due to inappropriate parameters. Moreover, the crossover method of genetic algorithm is introduced to reduce the influence that a Bayesian algorithm may fall into local optimization in the optimization process. In addition, the LSTM and XGBoost are used as detectors, and the final detection results are integrated according to the outputs of LSTM and XGBoost.

Chaofei Tang et al. [12] proposed a network IDS based on LightGBM and AE, in which the LightGBM algorithm was adopted for feature selection, and then AE was used as a classifier for training and detection. Yanqing Yang et al. [13] proposed a novel IDS model that combined improved conditional variational autoencoder (ICVAE) with a deep neural network (DNN). ICVAE was used to learn potential sparse representations automatically between features and classes in the data set. That can generate new attack samples according to the specified intrusion categories, balance the training data, and therefore increase the detection accuracy. Moreover, the ICVAE encoder was used to initialize the weight of DNN hidden layers to make forward and backward propagation easily. Xukui Li et al. [14] proposed an effective deep learning method based on AE and random forest (RF). In this model, RF was applied to select the best features. And then AP clustering algorithm is used to group the selected features into several feature subsets that was regarded as input of AE to calculate the RMSE. These RMSE values were the criterion to judge whether the network traffic was normal or not by K-means or GMM.
3. Materials and Methods

3.1. Overview of Approach. The proposed model is to detect network traffic and label them as benign or attack. Figure 1 shows the overview of our proposed model. In the model, this detection process can be divided into four stages. In the first stage, we divide the original data set into the LogAE data set and classifier data set, the former for training LogAE and the latter for training the classifier. In the next stage, we use the LogAE data set to train the LogAE model and store the model after finishing the training. In the third stage, we use the classifier data set to obtain the reconstructed data set by using the trained LogAE and then merge the two data sets. In the final stage, the combined data set is divided into a training data set and a testing data set to train and test the classifier, respectively.

3.2. Data Set. In this paper, the data sets we used are UNSW-NB15 and CICIDS2017.

UNSW-NB15 is created by the Australian Centre for Cyber Security (ACCS) [15]. Created features can be classified into five categories: flow features, basic features, content features, time features, and additional generated features [16]. Using the IXIA tool, 9 attack scenarios were created: Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms [17]. UNSW-NB15 data set has 257,673 data, of which 93,000 are normal samples and 164,673 are attack samples. Additionally, attributes in the UNSW-NB15 data set contain 44 features, including 42 attributes and 2 categories.

CICIDS2017 was proposed by the Canadian Cyber Security Institute to tease out the limitations of existing data sets and provide a realistic and reliable data set for intrusion detection [18]. It has a total of 2,830,743 records, each described by 80 network flow characteristics extracted from the generated network traffic using the CICFlowMeter tool. The CICIDS2017 data set contains benign and up-to-date common attacks, specific attack types including port scanning, botnets, web attacks, Heartbleed attacks, denial of service and distributed denial of service attacks, infiltration, SSH brute force, and FTP brute force attack. [19].

In this experiment, we divide the UNSW-NB15 data set and CICIDS2017 data set into two groups. One group is used to train the LogAE, which accounts for 30% of the data set, and the other is used to train the classifier, which accounts for 70% of the data set. In addition, the combined data that combines the reconstructed data with classifier data is divided into training data accounting for 60% and testing data accounting for 40%. Figure 2 shows the overview of data division and processing for UNSW-NB15. For any one input in classifier data, first, it is input to LogAE and then obtains the reconstruction data by LogAE. Next, the input data will be combined with the output data. This combination operation is to in a mathematical sense add each corresponding feature and then divide it by 2. In this regard, combined data is generated.

3.3. Logarithmic Autoencoder. Data normalization is always the main method of intrusion detection. The most commonly used methods are z-score normalization and min-max normalization. Data normalization can bring many benefits, such as improving the convergence speed of the model, improving the accuracy of the model, and preventing the explosion of model gradients in deep learning. For some features in intrusion detection datasets, the maximum value may be tens of thousands of times larger than the minimum value. This may cause operations such as data normalization to eliminate the influence of smaller data feature values on certain feature when processing larger data feature values, resulting in a decrease in the accuracy of the algorithm. This is also possible that the smaller data feature values in the large data feature range will lose data accuracy after conversion. For this shortcoming, Zhendong Wang et al. proposed logarithmic neural (LOGN), which has the ability to adjust the base value, so that it can have a more powerful mapping ability in the nonlinear region than the traditional linear operation [20]. Simply speaking, it can eliminate the need for data preprocessing steps, eliminate the interference of data normalization and other operations on the original data, and improve the accuracy of the model.

An autoencoder is a type of neural network. After training, it can try to copy the input to the output. In other words, it can make the output the same as the input. The network can be seen as composed of two parts: an encoder \( h = f(x) \) and a decoder \( r = g(h) \). The role of AE is to make \( x \) equal to \( g(f(x)) \). Theoretically, it is possible to design \( x = g(f(x)) \), but it is usually not done. The autoencoder should be designed so that it cannot learn to replicate perfectly. By imposing some constraints, the autoencoder can only replicate approximately, so it can learn the useful characteristics of the data.

The data reconstruction process of AE is to try to learn the hidden features of the input data and then try to reconstruct the data which is close to the original data according to the learned hidden features. In simple terms, the data reconstructed by the AE model can be maximally similar to the original data. This can also show that the reconstructed data set can reflect the importance of each
feature in the original data set to a certain extent. In other words, the reconstructed data based on hidden features can reflect the contribution of the features in the original dataset to the reconstruction process.

In this paper, we establish logarithmic autoencoder that combines logarithmic neural with autoencoder, named LogAE, which is described in Figure 3. LogAE mainly includes three parts, namely logarithmic layer, AE, and exponential layer. The logarithmic layer consists of LOGN, which is of the same order of magnitude as the input layer and has a one-to-one correspondence. The AE consists of a five-layer fully connected neural network, which aims to
learn the hidden features of the transformed data and then reconstruct the data. The exponential layer corresponds to the logarithmic layer, which transforms the data reconstructed by the AE based on the logarithmic base of the logarithmic layer.

In the LogAE, the loss function is defined as $\text{MAE} = \frac{1}{MN} \sum_{j=1}^{M} \sum_{k=1}^{N} |x_{jk} - x'_{jk}|$, and the logarithmic layer is defined as $x' = \ln(x + 1)/W - B$, where $W$ represents the logarithmic weight and $B$ is the bias of the logarithmic layer. Since there are certain values in the training data that are particularly large, we set $W = e^{3\times W}$.

In a nutshell, in our proposed module, the logarithmic layer is designed to learn mapping rules similar to data normalization, and the exponential layer performs the inverse operation according to the rules learned by the logarithmic layer. Between the logarithmic layer and the exponential layer, a common stacked AE is built. For the UNSW-NB15 data set, we compare different dimensions of hidden layers (see Table 1) and finally choose three layers as the dimensions of the hidden layer and the dimensions is (30, 15, 30). Similarly, for the CICIDS2017 data set, we compare different dimensions of hidden layers (see Table 2) and finally choose the best one that has the lowest MAE values.

### 3.4. Detection Classifier

Before introducing eXtreme Gradient Boosting (XGBoost), we first introduce the gradient boosting decision tree (GBDT). In a nutshell, GBDT accumulates the results of the classification and regression tree (CART) to make the conclusion. The core is that each tree learns the residual between the predicted value and the true value of all previous CART results. However, GBDT needs to traverse the entire data set multiple times at each iteration. If all data is loaded into memory, the size of the data will be limited; otherwise, repeatedly read and write operations will consume a lot of time. Therefore, when faced with massive data and high-dimensional data, GBDT cannot meet its needs. In order to overcome the difficulty of GBDT in dealing with large samples and high-dimensional data, XGBoost was born.

XGBoost was proposed by Tianqi Chen et al. [5]. It is an optimized distributed gradient enhancement library that implements machine learning algorithms under the gradient boosting framework, aiming to achieve high efficiency, flexibility, and portability. Compared with GBDT, XGBoost has several improvements: (1) the objective function in the algorithm is raised from the first order to the second order, and the algorithm supports a custom loss function; (2) the L2 regularization term of the number of leaf nodes and leaf weights is added, which reduces the variance of the model and helps prevent overfitting; (3) the base learner supports CART and linear classifiers; (4) the presorted data set is stored in a sparse matrix storage format so that the algorithm can use the block structure repeatedly and can more accurately find the data separation point, which greatly reduces the amount of calculation; and (5) sparsity-aware algorithms that automatically handle default values are added, and by calculating the gain of samples with default values classified as left and right branches and selecting the branch with the largest gain, the samples with default values are automatically divided; (6) column sampling is introduced, which

---

**Table 1: Comparison of AE structures for UNSW-NB15.**

| Hidden layer | MAE    |
|--------------|--------|
| (15)         | 3,015,547.5 |
| (30)         | 2,719,259    |
| (30, 15, 30) | 1,674,973.37 |
| (30, 20, 30) | 1,915,428    |
| (30, 15, 8, 15, 30) | 18,525,644 |
| (30, 20, 10, 20, 30) | 35,987,688   |

**Table 2: Comparison of AE structures for CICIDS2017.**

| Hidden layer | MAE    |
|--------------|--------|
| (40)         | 487,718.4062 |
| (20)         | 252,337.8594 |
| (40, 20, 40) | 93,488.8203  |
| (40, 30, 40) | 250,396.8281 |
| (40, 20, 10, 20, 40) | 383,536.5312 |
| (40, 30, 20, 30, 40) | 275,275.9688 |

---

![Figure 3: Structure of the logarithmic autoencoder.](image-url)
draws on the sampling algorithm of random forest, to reduce overfitting and reduce computation; and (7) shrinkage method is introduced. In each iteration, the weight of each leaf node of the tree is multiplied by the reduction weight, which weakens the influence of each tree and gives more room for optimization to the following trees.

4. Experiments and Results

4.1. Simulation Experiment Configuration. In terms of the verification of simulation results, this paper conducted two sets of comparative experiments on the mainstream data sets UNSW-NB15 and CICIDS2017 in the field of network intrusion detection. The first set of experiments is an ablation experiment, which mainly compares the effects of each module on performance, and the second group compares the algorithm with other existing new intrusion detection algorithms to evaluate the performance of the algorithm more comprehensively. The simulation experiment in this paper uses Python 3.6 for programming simulation under the Windows environment. The simulation environment is shown in Table 3.

4.2. Performance Metrics. To evaluate our proposed model, we use recall, precision, F1-score, and accuracy as the primary metrics, which can be implemented using Python’s scikit-learn machine learning library. Accuracy is the percentage of all correctly predicted samples in the total sample. Precision is that for our prediction results, it indicates how many of the samples whose predictions are positive are really positive samples, while recall is that for our original sample, it indicates how many positive examples in the sample are predicted correctly. F1-score is the weighted harmonic average of precision and recall.

The statistical standards are defined as follows:

---

Table 3: The simulation environment.

| Project                | Environmental parameters |
|------------------------|--------------------------|
| Python version         | 3.6                      |
| System environment     | Windows 10 x 64          |
| RAM                    | 16.0 GB                  |
| CPU                    | Intel® Xeon® CPU E3-1241 v3 @ 3.50 GHz |

Table 4: Ablation experiment: comparison between LogAE-XGBoost and XGBoost

| Data set      | Classifier   | Accuracy | Precision | Recall | F1-score |
|---------------|--------------|----------|-----------|--------|----------|
| UNSW-NB15     | LogAE-XGBoost| 0.9511   | 0.9549    | 0.9743 | 0.9645   |
|               | XGBoost      | 0.9023   | 0.9874    | 0.8676 | 0.9236   |
| CICIDS2017    | LogAE-XGBoost| 0.9992   | 0.9971    | 0.9986 | 0.9979   |
|               | XGBoost      | 0.9943   | 0.9922    | 0.9930 | 0.9925   |

Table 5: Comparison of the proposed model and other excellent classifiers for UNSW-NB15.

| Classifier              | Accuracy | Precision | Recall | F1-score | Time(s) |
|-------------------------|----------|-----------|--------|----------|---------|
| MSCNN [21]              | 0.9140   | -         | -      | -        | 571     |
| MSCNN-LSTM [21]         | 0.9560   | -         | -      | -        | 1,060   |
| SaE-ELM-Ca [22]         | 0.8917   | 0.8079    | 0.9958 | 0.8920   | 2.168   |
| SDAE-ELM3 [23]          | 0.7238   | 0.6994    | 0.8742 | 0.7771   | 5.943   |
| XGBoost-DNN [8]         | 0.9950   | 0.9945    | 0.9942 | 0.9952   | 4.200   |
| LogAE-XGBoost           | 0.9511   | 0.9549    | 0.9743 | 0.9645   | 132     |

Table 6: Comparison of the proposed model and other excellent classifiers for CICIDS2017.

| Classifier              | Accuracy | Precision | Recall | F1-score | Time(s) |
|-------------------------|----------|-----------|--------|----------|---------|
| NB-SVM [24]             | 0.9892   | 0.9946    | 0.9700 | 0.9821   | —       |
| T-SNERF [25]            | 0.9978   | 0.9980    | 0.9980 | 0.9980   | —       |
| MTH-IDS [26]            | 0.9989   | 0.9981    | 0.9160 | 0.9989   | 478.2   |
| LMDRT-SVM2 [27]         | 0.9928   | 0.9916    | 0.9939 | 0.9927   | —       |
| DT + rules-based model [28] | 0.9666 | 0.9447    | 0.9886 | 0.9662   | 160.07  |
| KNN [18]                | 0.9630   | 0.9620    | 0.9370 | 0.9630   | 15,243.6|
| RF [18]                 | 0.9882   | 0.9880    | 0.9985 | 0.9880   | 1,848.3 |
| LogAE-XGBoost           | 0.9992   | 0.9971    | 0.9986 | 0.9979   | 1,092.35|

It is used to distinguish the metrics of our proposed model from those of other models.
Two sets of experiments are performed to demonstrate the effectiveness and scalability of the model we proposed. One group is ablation experiments, which prove that LogAE improves the performance of XGBoost, and the other group is comparative experiments that compare with other advanced models to prove the superior performance of the model we propose. The UNSW-NB15 data set, based on a comparison of the above two sets of experiments, shows that although our proposed model is lower than a few IDS models, it also has high accuracy, recall rate, accuracy, and F1-score, which are 95.11%, 95.45%, 97.43%, and 96.45%, respectively. And it is much lower than most IDS models in runtime. Compared with the fastest running time model, the running time of our proposed model is 132 seconds, which is around one-fifth of its running time. For the CICIDS2017 data set, based on a comparison of the above two sets of experiments, it is shown that the model we propose has higher accuracy, recall, accuracy, and F1 indicators, which are 99.92%, 99.71%, 99.86%, and 99.79%, respectively. In terms of runtime, our proposed model is 1,092.35 seconds, which is in the middle level. Overall, we conclude that LogAE-XGBoost offers best-in-class predictive performance.

Data Availability

The data used to support the findings of this study have been deposited in https://github.com/Xuwenfeng-GUET/LogAE-XGBoost

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant nos. 62162015 and 61762026 and in part by the Innovation Project of GUET Graduate Education under Grant no. 2021YCXS066.

References

[1] T. Hamed, R. Dara, and S. C. Kremer, “Network intrusion detection system based on recursive feature addition and bigram technique,” Computers & Security, vol. 73, pp. 137–155, 2018.
[2] P.-F. Marteau, “Sequence covering for efficient host-based intrusion detection,” IEEE Transactions on Information Forensics and Security, vol. 14, no. 4, pp. 994–1006, 2019.
[3] J. M. Luna, E. D. Gennatas, L. H. Ungar et al., “Building more accurate decision trees with the additive tree,” Proceedings of the National Academy of Sciences, vol. 116, no. 40, pp. 19887–19893, 2019.
[4] M. Mohammadi, T. A. Rashid, S. H. T. Karim et al., “A comprehensive survey and taxonomy of the SVM-based intrusion detection systems,” Journal of Network and Computer Applications, vol. 178, Article ID 102983, 2021.
[5] T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 2016.

[6] D. R. C. Canêdo and A. R. S. R. Romariz, "Intrusion detection system in ad hoc networks with artificial neural networks and algorithm K-means," IEEE Latin America Transactions, vol. 17, no. 7, pp. 1109–1115, 2019.

[7] K. Sadaf and J. Sultana, "Intrusion detection based on autoencoder and isolation forest in fog computing," IEEE Access, vol. 8, pp. 167059–167068, 2020.

[8] S. T. Ikram, A. K. Cherukuri, B. Poorva et al., "Anomaly detection using XGBoost ensemble of deep neural network models," Cybern. and Info. Techn. vol. 21, no. 3, pp. 175–188, 2021.

[9] H. Jiang, Z. He, G. Ye, and H. Zhang, "Network intrusion detection based on PSO-Xgboost model," IEEE Access, vol. 8, pp. 58392–58401, 2020.

[10] C. Karthikraja, J. Senthilkumar, R. Harirahan, G. U. Devi, Y. Suresh, and V. Mohanraj, "An empirical intrusion detection system based on XGBoost and bidirectional long-short term model for 5G and other telecommunication technologies," Computational Intelligence, pp. 1–16, 2022.

[11] C. Song, Y. Sun, G. Han, J. P. C. Rodrigues, and P. C. Rodrigues, "Intrusion detection based on hybrid classifiers for smart grid," Computers & Electrical Engineering, vol. 93, Article ID 107212, 2021.

[12] C. Tang, N. Luktarhan, and Y. Zhao, "An efficient intrusion detection method based on LightGBM and autoencoder," Symmetry, vol. 12, no. 9, p. 1458, 2020.

[13] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational AutoEncoder and deep neural network," Sensors, vol. 19, no. 11, p. 2528, 2019.

[14] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building Auto-Encoder Intrusion Detection System based on random forest feature selection," Computers & Security, vol. 95, Article ID 101851, 2020.

[15] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in Proceedings of the Military Communications and Information Systems Conference (MiCIS), IEEE, Canberra, Australia, November 2015.

[16] X. Larriva-Novô, V. A. Villagrà, M. Vega-Barbas, D. Rivera, and M. S. Rodrigo, "An IoT-focused intrusion detection system Approach based on preprocessing characterization for cybersecurity datasets," Sensors, vol. 21, no. 2, 2021.

[17] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: datasets and comparative study," Computer Networks, vol. 188, Article ID 107840, 2021.

[18] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proceedings of the 4th International Conference on Information Systems Security and Privacy, Funchal, Portugal, January 2018.

[19] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," International Journal of Engineering & Technology, vol. 7, no. 1/1, pp. 479–482, 2018.

[20] Z. Wang, Z. Xu, D. He, and S. Chan, "Deep logarithmic neural network for Internet intrusion detection," Soft Computing, vol. 25, no. 15, pp. 10129–10152, 2021.