1-1-2022

Flexible and Robust Real-Time Intrusion Detection Systems to Network Dynamics

Kicho Yu
Northeastern University

Khanh Nguyen
San Jose State University, khanh.nguyen@sjsu.edu

Younghee Park
San Jose State University, younghee.park@sjsu.edu

Recommended Citation

Kicho Yu, Khanh Nguyen, and Younghee Park. "Flexible and Robust Real-Time Intrusion Detection Systems to Network Dynamics" *IEEE Access* (2022): 98959-98969. https://doi.org/10.1109/ACCESS.2022.3199375

Received 13 June 2022, accepted 7 July 2022, date of publication 17 August 2022, date of current version 23 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3199375

RESEARCH ARTICLE

Flexible and Robust Real-Time Intrusion Detection Systems to Network Dynamics

KICHO YU[1], KHANH NGUYEN[2], AND YOUNGHEE PARK[2], (Member, IEEE)
[1]Khoury College of Computer Science, Northeastern University, Boston, MA 02115, USA
[2]College of Engineering, San José State University, San Jose, CA 95192, USA

Corresponding author: Younghee Park (younghee.park@sjsu.edu)

ABSTRACT Deep learning-based intrusion detection systems have advanced due to their technological innovations such as high accuracy, automation, and scalability to develop an effective network intrusion detection system (NIDS). However, most of the previous research has focused on model generation through intensive analysis of feature engineering instead of considering real environments. They have limitations to applying the previous methods for a real network environment to detect real-time network attacks. In this paper, we propose a new flexible and robust NIDS based on Recurrent Neural Network (RNN) with a multi-classifier to generate a detection model in real time. The proposed system adaptively and intelligently adjusts the generated model with given system parameters that can be used as security parameters to defend against the attacker's obfuscation techniques in real time. In the experimental results, the proposed system detects network attacks with a high accuracy and high-speed model upgrade in real-time while showing robustness under an attack.

INDEX TERMS Long short-term memory, network intrusion detection system, recurrent neural network, real-time data analysis.

I. INTRODUCTION

Deep learning has been popularly applied in various applications and solutions in diverse fields, including image processing and autonomous driving. In addition, deep learning techniques have provided many benefits to developing network intrusion detection system (NIDS) due to automation and high accuracy. Without human intervention, NIDS can detect (un)known network attacks through intensive data analysis based on historical attack data. Therefore, deep learning-based NIDS is one of the most important defense methods to automatically monitor network behavior and to detect abnormal behavior based on the built-in attack models through automatic feature engineering.

Many deep learning-based IDSes (DL-IDS) have been proposed for a decade to improve the attack detection techniques due to advantages such as automatic feature generation, effectiveness and scalability [1], [2], [3], [4], [5], [7], [8], [9], [10], [17], [23]. Many deep learning methods, such as CNN, GAN, and Autoencoder, have been popularly utilized for the development of NIDS [21], [22], [34], [35], [36]. Zhang *et al.* proposed both SMOTE and GAN for NIDS [27], [28], [29]. Several research studies applied CNN based on LSTM for an intrusion detection that is appropriate for two-dimension data [34], [35], [36]. Yuan *et al.* developed a DDOS detection method [34] and Radford *et al.* proposed a detection method to evaluate the sequence features in two directions after modifying the LSTM technique [35]. Wang *et al.* proposed the modified CNN to learn spatial features and LSTM to learn time features [36]. In addition, Zeng *et al.* proposed a payload detection method with multiple deep learning models: CNN, LSTM, and a stacked autoencoder [21]. Yu *et al.* used a convolutional autoencoder to extract payload features [22]. Rigaki *et al.* used GAN to improve the malware detection, because adversarial learning like GAN enhances the robustness of IDS [23].

However, despite the trustworthiness of the DL-IDS, challenges remain concerning the development of real-time intelligent IDS by adapting to network dynamics. The network traffic patterns are often changed due to various conditions and network environments. In general, attackers try to hide their actual features and avoid detection by obfuscating their

The associate editor coordinating the review of this manuscript and approving it for publication was Amin Zehtabian.

behavior. The previous DL-IDS that fully relies on historical training data for attack model generation cannot adapt to the real-time network behavior by reflecting the change of network features. Thus, we need to work on several important issues: (i) data dependency, (2) model dependence, and (3) impracticality. In other words, the current DL-IDS systems have three main weaknesses: first, they cannot update built-in attack models in real-time by including new network data and new network features. Second, the pre-built-in attack models cannot be improved by reflecting current network behavior. Third, due to these two limitations, the previous DL-IDS cannot show high accuracy in a real environment to detect real-time attacks since they cannot adjust their system to new network dynamics in real time. Therefore, it is important to address the challenges by developing a new NIDS by considering network dynamics and real-time characteristics in networks.

This paper proposed a flexible and robust NIDS by using deep learning while updating the built-in attack models in real time by considering current network behavior and performance. The proposed system utilizes Recurrent Neural Network (RNN) with a multi-classifier in order to randomly select new data sets depending on system parameters in the system. It has two different approaches: the best-effort approach and the adaptive feature-engineering approach. The best-effort approach aims to upgrade the pre-built-in attack models by training data sets by randomly selecting a new data set in real time under a given random traffic size and time. The adaptive feature-engineering approach also updates the pre-built-in models through feature engineering along with the first method, the best-effort approach. In other words, the second method replaces the current model with the new model by updating the attack model with the new feature sets depending on the current network dynamics and system performance in real time. Therefore, the proposed method can detect network attacks effectively since it adapts to the current network environment. The proposed system randomly selects high-quality new data while deleting ambiguous data sets within a given random time. Due to the random time and data selection in order to upgrade the current model, attackers cannot disturb the proposed process even though attackers try to obfuscate real-time traffic patterns. Through four different data sets in NIDS, such as NSL-KDD 99, Kyoto 2006, UNSW-NB15, and CIDDS, the proposed system presented high performance and robustness under attacks in real time.

The paper contributes to the following aspects in DL-NIDS. First, the proposed system first presents a real-time adaptive and robust NIDS based on deep learning. Second, the proposed system has random features to prevent attackers from obfuscating current traffic to interrupt the model generation in real time. Third, the paper explains the relationship between data sizes and model accuracy with diverse parameters in the system. Finally, we evaluate the proposed system by using large different data sets with different factors. We also demonstrate the robustness of the proposed model under attack.

The rest of the paper is organized as follows: Section II discusses the previous NIDS based on machine learning and deep learning. Section III presents our proposed system and Section IV shows our data sets and experimental results. Finally, we will conclude our work in Section VI while discussing our methods in different angles in Section V.

II. RELATED WORKS

Machine learning (ML) and deep learning (DL) techniques have been popularly adapted to develop intrusion detection systems (IDS) because of their high accuracy, automation, and no previous knowledge requirement [1], [2], [3], [4], [7], [8], [9], [10], [17]. IDS can be deployed at a single computer such as host-based intrusion detection system (HIDS) to many networks as network-based intrusion detection system (NIDS) [15], [16]. IDS can be categorized based on a detection method: signature-based detection method and anomaly-based detection method [15].

There are various kinds of machine learning-based IDS, since a machine learning can be applied to packet-based attack detection in IDS. Mayhew *et al.* proposed a packet parsing-based detection method based on SVM and K-means [18]. Hu *et al.* proposed a packet parsing-based detection method based on a fuzzy C-means to reduce the false alarm rate and the missed alarm rate [19]. Min *et al.* used a text-based CNN to detect attacks from payloads that provided content features [20]. Zeng *et al.* adopted different deep learning models (CNN, LSTM, and a stacked autoencoder) to extract features as a payload analysis [21]. Yu *et al.* trained a convolutional autoencoder model to extract payload features [22]. As adversarial learning enhances the robustness of IDS, Rigaki *et al.* used a GAN to improve the malware detection effect [23].

Machine learning can be applied to a feature engineering-based detection method in which common features are packet length, the proportion of TCP flags, and source byte [17]. Machine learning-based intrusion detection systems can be combined with SVM, decision tree, Naïve Bayes, and K-Means to increase accuracy or to accelerate the detection speed [24], [25], [26]. Ahmim *et al.* proposed a hierarchical decision tree method as a part of statistic-based feature detection methods [32]. Alseiari *et al.* applied K-Means to detect attacks in smart grid [33]. Moreover, deep learning-based detection learns features without previous knowledge. Potluri *et al.* proposed a CNN-based detection method because CNN is suitable to process 2-dimensional data, and they used that after converting the feature vectors into 2-dimensional images [27]. Zhang *et al.* used SMOTE to up-sample the minority classes such as User to Root attacks and Remote to User attacks to make the class balanced and then XGBoost to detect attacks [28]. Zhang *et al.* improved the aforementioned approach by GAN — adversarial learning — to improve accuracy in seven out of eight attack types [29]. Teng *et al.* proposed a detection method based on SVM by grouping traffic according to a protocol type such as TCP, UDP, and ICMP [30]. Ma *et al.* proposed a spectral clustering-based detection method after training with DNN [31].

Many research projects have also worked on sequence data to generate a model based on time-series data. Sequence feature-based detection for NIDS has been evolved from CNN and RNN with the LSTM model since the deep learning technique can consider sequence data to generate a model. Yuan *et al.* proposed a DDOS detection method based on LSTM and CNN [34]. Radford *et al.* proposed a session detection method based on a bi-LSTM, because bi-LSTM is suitable to lean the sequence features in two directions [35]. Wang *et al.* proposed hierarchical deep learning using a character-level CNN to learn spatial features and LSTM to learn time features [36].

Hybrid approaches usually achieved better accuracy because of the combination of two different methods. Some researchers combined rule-based methods and machine learning-based methods. Meng *et al.* proposed a KNN method to rank alerts, whereas McElwee *et al.* proposed a DNN to filter alarms from McAfee data [37], [38]. Other research focused on log data instead of network data by extracting important features from network and system log based on domain knowledge to discover anomalies [39], [40], [41]. Uwagbole *et al.* proposed an SQL-injection detection method for the Internet of Things (IoT) using SVM [42]. Vartouni *et al.* proposed a web attack detection method based on the isolate forest model [43].

III. OUR APPROACH

This section presents our approach to develop a flexible intrusion detection system to adapt to network dynamics over time by using RNN, as shown in Figure 1. The system architecture has three parts: data processing, data classification through multi-classifier, and RNN Modeling. To build the real-time robust IDS, the proposed system consists of two different methods to update the proposed system in real-time: (1) the best-effort approach and (2) the adaptive feature-engineering approach. The first method continues improving the trained model over time by adding a high-quality real-time data through an eclectic approach. The second method adjusts the existing model by adding extra feature sets based on the feature importance with the new data. The proposed method demonstrates the effectiveness and the robustness by upgrading the current attack models in real time by adjusting data and network features. The following sections will be discussed in detail.

A. DATA CLASSIFICATION BY A MULTI-CLASSIFIER

The proposed system first performs data processing and data classification based on the multi-classifier in Figure 1. The data processing first performs data cleansing by using archived historical data for a model generation and real-time incoming data to update the generated model in real time. In other word, the data processing is where we load datasets, clean them, and balance them in terms of the binary dependent variables.

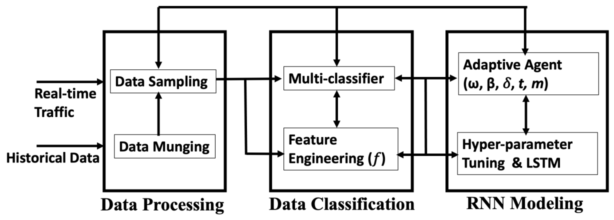


FIGURE 1. Real-time intrusion detection system architecture.

After the data cleaning and sampling, the proposed system utilizes the Random Forest algorithm as a multi-classifier to select the best quality of data and to evaluate the feature importance. Random Forest consists of multiple decision trees as an ensemble classifier through bagging which is to count and average the votes from each decision tree [44]. Bagging is also called as bootstrap aggregation and it reduces the variance which is a proxy for a consistency. The vote can be mathematically expressed as

$$\hat{C}(x) = majority\ vote\ \{\hat{C}_i(x)\}_{i=1}^{n} \tag{1}$$

where we have a total n trees and $\hat{C}_i(x)$ is the classification of ith random forest tree [6].

As a byproduct of Random Forest, in which feature importance is generated. It is a list of features and how quantitatively important they are in Random Forest decision making. It is calculated using a normalization on Gini Impurity.

$$\sum_{i=1}^{C} f_i(1 - f_i) \tag{2}$$

where f_i is the frequency of label i at a node and C is the number of unique labels. When a tree sprouts a branch, the improvement in the split-criterion is the importance measure attributed to the splitting variable, and is accumulated over all the trees in the forest separately for each variable [6].

Based on the results of the multi-classifier by using the two equations (1) and (2), the system collects the most promising datasets to be used for input for RNN in the next step. It also selects feature sets based on the outcomes of the feature importance, as we demonstrated in the evaluation section. Our previous work proposed a multi-classifier by exploiting various machine learning techniques to exclude ambiguous data from the training data for high accuracy [3], [4].

As discussed in this Section III-A, the multi-classifier outperforms in data classification by detecting outliers, which results in decreasing system performance based on our previous research outcomes [3] and [4]. In addition, the multi-classifier showed higher speed to perform data classification than deep learning algorithms due to many hidden layers.

B. ADAPTIVE RNN MODELING

As shown in Figure 1, the data processing selects and cleans historical data or real-time data through data sampling to balance datasets. Then, the multi-classifier with the results of the feature engineering (f) collects high-quality data after excluding ambiguous data as explained in Section III-A. The feature engineering is where we perform a feature transformation to convert string features into a numerical feature, use a multi-classifier to classify features to detect malicious traffic from benign traffic, and then apply hyperparameter tuning on the RNN model to use in the last process. The continuous data processing is where we imitate real-time data processing from the datasets. Lastly, the RNN modeling is where we split data into initial inputs and sequential inputs, control our environment with threshold, and run an RNN model with the LSTM capability.

Recurrent Neural Network (RNN) is a type of an artificial neural networks that is used for sequential or temporal data. The basic RNN has the following architecture in Figure 2, where x_i is an ith input, y_i is an ith output, W is a weight matrix, and h_i is an ith hidden layer with an activation function. The proposed system uses Sigmoid as an activation function because that leads the highest accuracy as discussed in Section IV-C. It also uses LSTM instead of a simple RNN to mitigate RNN's innate vanishing gradient problem.

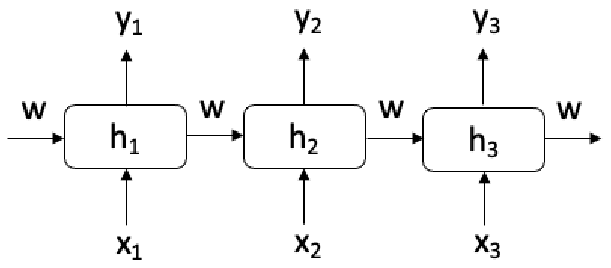


FIGURE 2. A Recurrent Neural Network (RNN).

As explained in the previous subsection, an RNN model has an innate long-term memory loss, due to multiplicative gradient that can be exponentially decreasing with respect to the number of layers. We use LSTM to preclude the early stage of memory loss to update training. The standard LSTM has input gates and output gates. The net input and the activation with in_j on the j-th memory cell are

$$net_{in_j}(t) = \sum_u w_{in_j u} y^u(t-1)$$
$$y^{in_j}(t) = f_{in_j}(net_{in_j}(t)) \tag{3}$$

where y is an activation function of the input [10].

The net output and the activation of out_j on the j-th memory cell are

$$net_{out_j}(t) = \sum_u w_{out_j u} y^u(t-1)$$
$$y^{out_j}(t) = f_{out_j}(net_{out_j}(t)) \tag{4}$$

where y is an activation function of the output [10].

In this paper, we propose a new network intrusion detection system by utilizing the RNN model with the multi-classifier. The proposed system has different system parameters, such as a random time (Δt), a window size (ϖ), and a block size (β), to build a model in real time. The proposed system collects data at a randomly selected time (Δt). The collected data size is determined by the two system parameters: a window size (ϖ) and a block size (β). The window size is the data size to generate a model, and the block size is the data to be used for model upgrades in real time.

To improve the RNN models in real time, this paper proposes two approaches: (1) the best-effort approach and (2) the adaptive feature-engineering approach, as described in the following.

1) THE BEST-EFFORT APPROACH

Given a random time (Δt), a window size (ϖ), and a block size (β), the proposed system keeps improving the current model when the system achieves better system performance (m) as an accuracy. For example, at a random time, the system processes a set of data based on the value of the window size to generate the first model. The system updates the current model with the new model by regenerating the new model with the original data sets (i.e. the amount of the window size) and additional data according to the block size (β). Based on the result of the multi-classifier according to the system parameter values, the input data will be provided to the input gates at the RNN modeling as in Eq 5 and 5. Those equations are where j-th memory cell has an input gate in_j and an output gate out_j. The input gate's activation at time t and the output gate's activation at time t are $y^{in_j}(t)$ and $y^{out_j}(t)$ respectively [10].

Unlike the standard LSTM, our LSTM has a threshold value δ that a metric m compares with. For example, if we choose a metric m as an accuracy, then a threshold value δ is the best by-far accuracy. This makes our LSTM equations for input gate and output gate as follows respectively.

$$net_{in_j}(t) = \sum_u w_{in_j u} y^u(t-1)$$
$$y^{in_j}(t) = \begin{cases} f_{in_j}(net_{in_j}(t-1)) & m \geq \delta \\ f_{in_j}(net_{in_j}(t-1)) & m < \delta \end{cases}$$
$$net_{out_j}(t) = \sum_u w_{out_j u} y^u(t-1) \tag{5}$$
$$y^{out_j}(t) = \begin{cases} f_{out_j}(net_{out_j}(t-1)) & m \geq \delta \\ f_{out_j}(net_{out_j}(t-1)) & m < \delta \end{cases} \tag{6}$$

In this way, our model can be selectively updated based on the threshold δ.

2) THE ADAPTIVE FEATURE-ENGINEERING APPROACH

The approach generates a new model based on the updated new feature sets by considering the system parameters that are used for the best-effort approach. In other words, the adaptive feature-engineering approach changed the feature sets on the top of the best-effort approach. In detail, based on the aforementioned δ threshold, our LSTM adaptively updates features seen as $f\prime$ in Equation (7) and (8). The list of features was selected from Random Forest's Feature Importance as explained in Section III-A. When a metric such as an accuracy (or a recall) exceeds the δ threshold, we update the current feature sets by adding new features or deleting old features. If it does not exceed, then our LSTM is the same as a regular LSTM.

$$net_{in_j}(t) = \sum_u w_{in_j u} y^u(t-1)$$
$$y^{in_j}(t) = \begin{cases} f\prime_{in_j}(net_{in_j}(t-1)) & m \geq \delta \\ f_{in_j}(net_{in_j}(t-1)) & m < \delta \end{cases} \tag{7}$$

The net output and the activation of out_j are

$$net_{out_j}(t) = \sum_u w_{out_j u} y^u(t-1)$$
$$y^{out_j}(t) = \begin{cases} f\prime_{out_j}(net_{out_j}(t-1)) & m \geq \delta \\ f_{out_j}(net_{out_j}(t-1)) & m < \delta \end{cases} \tag{8}$$

IV. EVALUATION

This section presents our experiments setup and results to evaluate our proposed system by using four different public datasets: NSL-KDD 99, UNSW-NB15, Kyoto 2006, and CIDDS. We used MacBook Pro 2019 2.4 GHz 8-Core Intel Core i9 64 GB 2667 MHz DDR4 to measure accuracy, and True Positive and False Positive Rates. First, we explained each dataset with the feature sets and now we will present our experimental results to show the effectiveness and the robustness of the proposed system while computing the area under curve (AUC) to characterize the performance of the proposed system based on the ROC (receiver operating characteristic) curve.

A. DATASETS

We summarize four different public datasets (NSL-KDD 99, UNSW-NB15, Kyoto 2006, and CIDDS) to evaluate our proposed system as follows.

1) NSL-KDD 99 [11]

NSL-KDD 99 is an improvement of KDDCUP'99 dataset. It has no duplicate records in the training dataset. This prevents a model from having a high bias toward frequent records. There are 42 features along with 1 feature called "class" which explicitly explain the data packet being malicious or benign. The attacks are in 4 categories: Denial of service (DoS), user to root (U2R), remote to local (R2L), and probing (PROBE).

2) UNSW-NB15 [12]

Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) created this dataset. It contains hybrids of the modern normal and contemporary attack patterns actively collected on network traffic. It has 45 features including two columns that specify the type of data packet and the attack category.

3) Kyoto 2006 [13]

The dataset contains 24 statistical features: both numerical and categorical features. One column indicates the type of packet: normal, known attack, or unknown attack. In our data analysis, we dropped packets with an unknown attack and downsampled. Specifically, we downsampled 2,613,808 known attack packets to 130,742 normal packets. The Kyoto dataset includes important features required for detecting an intrusion in a system such as source/destination bytes, flag status, the duration of the connection, IP addresses. These features are recognized and selected using feature selection techniques. The final attributes are chosen from the association data packets with the existing features, which are used for intrusion detection in real time. The features selected are duration, source bytes, flag, Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, attack label.

4) CIDDS [14]

CIDDS-001 (Coburg Network Intrusion Detection Dataset) was created for the purpose of evaluation of Anomaly-based Network IDS. The dataset contains 14 features in total including a column that mentions if the data packet is attacker, victim, or normal. It includes both numerical and categorical features. Due to the size of the data and our computing power, we have trained our models using CIDDS-001-internal-week1 file.

B. RESULTS OF THE MULTI-CLASSIFIER

As we discussed in Section III-A, the multi-classifier can improve system accuracy with the quick data processing time compared to deep learning techniques while deleting ambiguous data from the collected data. Figure 3, 4, 5, and 6 showed ROC AUC from different machine learning models: Logistic Regression, Decision Tree, KNN, Random Forest, Multilayer Perceptron, Gaussian Naïve Bayes, and Gradient Boost.

The proposed system utilized the Random Forest algorithm to achieve our data classification goal since it showed the best accuracy for the four different datasets, as shown in the benchmark results in this experiment. Note that the proposed system can also utilize more than one machine learning algorithm to create an ensemble method for the solution of the data classification problem, as presented in our previous work.

In addition, the Random Forest algorithm generates feature importance as explained in Section III-A. As shown in Figure 1, we used the top ten features from all the available features based on the feature importance from the Random Forest algorithm experiments. The importance values of each feature for each dataset ranges from around 0.03 to 0.35 bits as the entropy outcomes. NSL-KDD 99 has 43 features and its top features are src_bytes, dst_bytes, and difficulty_level. UNSW-NB 15 has 45 features and its top features are attack_cat, sttl, and ct_state_ttl. Kyoto 2006 has

TABLE 1. List of the best ten features for each dataset.

Rank	NSL-KDD 99 Feature	UNSW-NB 15 Feature	Kyoto 2006 Feature	CIDDS Feature
1)	src_bytes	attack_cat	destination_ip_address	attackID
2)	dst_bytes	sttl	destination_port_number	attackType
3)	difficulty_level	ct_state_ttl	dst_host_srv_count	attackDescription
4)	flag	id	flag	Packets
5)	same_srv_rate	sload	service	Flags
6)	dst_host_srv_count	rate	dst_bytes	Duration
7)	diff_srv_rate	sbytes	dst_host_count	Bytes
8)	logged_in	dload	dst_host_srv_serror_rate	Dst IP Addr
9)	count	smean	count	Src IP Addr
10)	dst_host_diff_srv_rate	ct_srv_dst	src_bytes	Tos

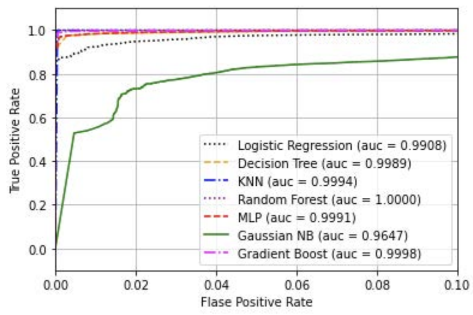


FIGURE 3. Performance evaluation with NSL-KDD 99 dataset.

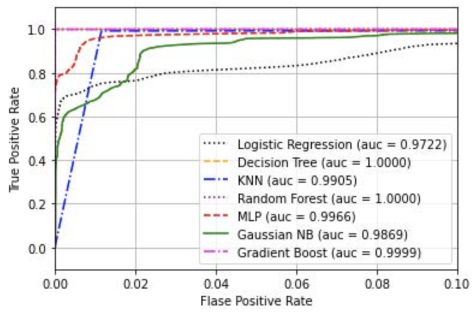


FIGURE 5. Performance evaluation with kyoto 2006 dataset.

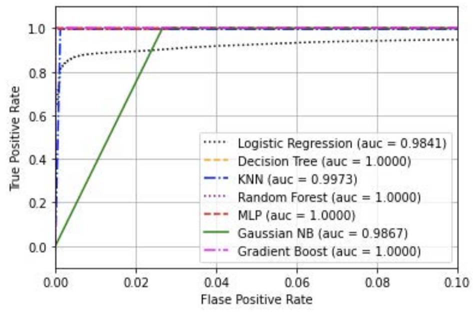


FIGURE 4. Performance evaluation with UNSW-NB15 dataset.

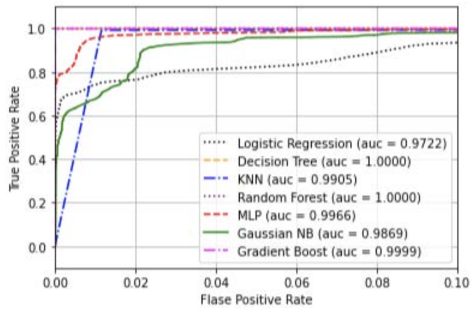


FIGURE 6. Performance evaluation with CIDDS dataset.

24 features and its top features are destination_ip_address, destination_port_number, and dst_host_srv_count. CIDDS has 16 features and its top features are attackID, attackType, and attackDescription.

C. RESULTS OF HYPERPARAMETER TUNING

A hyperparameter tuning is an important part of improving a deep learning model. This paper performed a hyperparameter tuning onto activation function, learning rate, and dropout rate.

The result of hyperparameter tuning is as shown in Table 2. This paper tested activation functions first. We tested Relu, Sigmoid, and Softmax. Sigmoid with binary cross-entropy shows the highest accuracy. For example, we achieved accuracies of 90.82%, 90.99%, and 91.71% for 80K, 100K, and 120K window size, respectively, in the Kyoto 2006 dataset.

Once we found that Sigmoid activation function leads with the highest accuracy, we tested learning rate and dropout rate. We set up the dropout rates as 0.05, 0.1, and 0.15, and the learning rates as 1, 0.5, 0.1, and 0.05.

Based on those experiments and as seen from Table 2 on page 98965, we conclude that the dropout rate is optimal at 0.15 for all 4 datasets, and the learning rate is optimal around 0.1 or 0.05, depending on the dataset. Overall, a learning rate provides more weights than a dropout rate. In other words, a learning rate is metric-elastic, whereas a dropout percentage is metric-inelastic.

D. IMPACT OF WINDOW SIZE AND BLOCK SIZE

This paper has differentiated the training size (i.e. window size, ϖ) into three categories for an experiment: 50K, 100K, and 150K traces: 50K traces for RNN1($\varpi = 50$K), 100K traces for RNN2($\varpi = 100$K), and 150K traces for RNN3($\varpi = 150$K), respectively, in the experimental results. In other words, the proposed system generated three different models based on the three different windows sizes. After generating the first model for each, given a time ($\Delta\ t$), the proposed system updates the generated model with the two different block sizes, $\beta = 20$K or 40K traces. The block size is the amount of the new real-time data that we feed into the

TABLE 2. Kyoto hyperparameter tuning (accuracy).

Window Size	dropout	lr=0.1	lr=0.05
80K	0.05	89.78%	95.67%
80K	0.1	90.41%	90.39%
80K	0.15	90.89%	89.77%
100K	0.05	90.26%	90.15%
100K	0.1	90.52%	90.25%
100K	0.15	90.34%	90.72%
120K	0.05	89.71%	90.31%
120K	0.1	90.47%	90.64%
120K	0.15	90.49%	95.92%

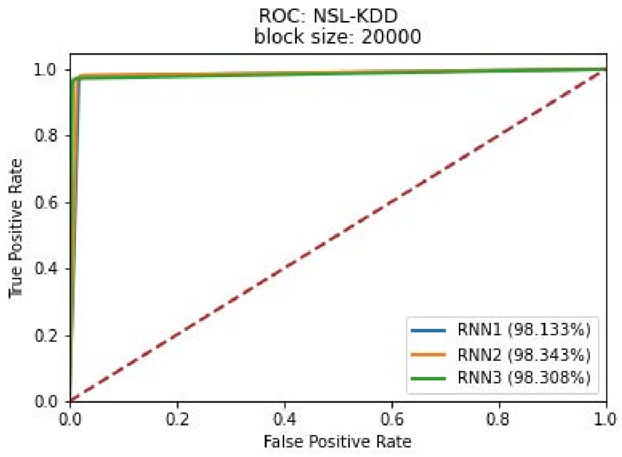


FIGURE 7. Performance evaluation with NSL-KDD 99 dataset ($\beta = 20K$).

proposed system to regenerate the new model. Since we used historical archival datasets, we simulated the real-time data feed by adding random new data for each dataset. The system set up with a threshold (δ) as a current accuracy. For example, for RNN1($\varpi = 50K$) and the window size $\beta = 20K$, the system first creates an attack model to detect network attacks in real time. After that, when the system keeps improving the model with the highest accuracy than the threshold (i.e. the current accuracy), the proposed system replaces the current model with the new model by combining the original 50K dataset with additional 20K dataset.

In NSL-KDD 99 dataset, the performance of the RNN2 case is slightly better than the other two cases (RNN1 and RNN3) for both different block sizes (i.e. $\beta = 20K$ or 40K traces). Under the 20K block size, RNN2 showed 97.710% True Positive Rate and 1.908% False Positive Rate. That True Positive Rate is slightly higher than 97.390% and 97.584% from RNN1 and RNN3 respectively. Under the 40K block size, RNN2 showed 98.019% True Positive Rate and 5.230% False Positive Rate. That True Positive Rate is lightly better than 97.739% and 97.770% from RNN1 and RNN3 respectively.

In UNSW-NB15 dataset, the performance from RNN3 performs relatively the best than other cases for both block sizes, but the difference between RNN1 and RNN2 is minimal. Under the 40K block size, RNN3 showed almost 99.991% True Positive Rates, whereas RNN1 and RNN2 showed 98.512% and 99.748% True Positive Rates respectively.

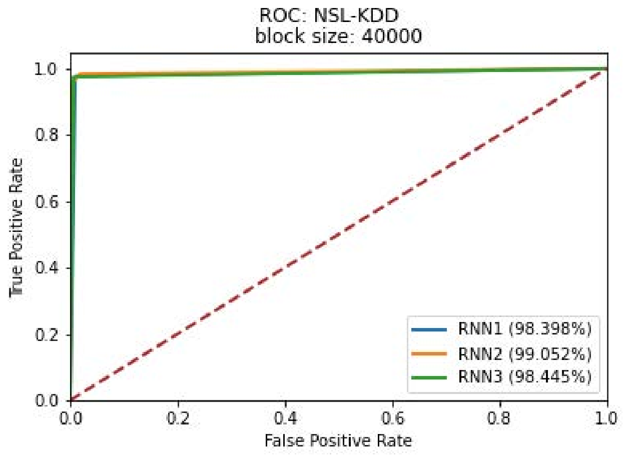


FIGURE 8. Performance evaluation with NSL-KDD 99 dataset ($\beta = 40K$).

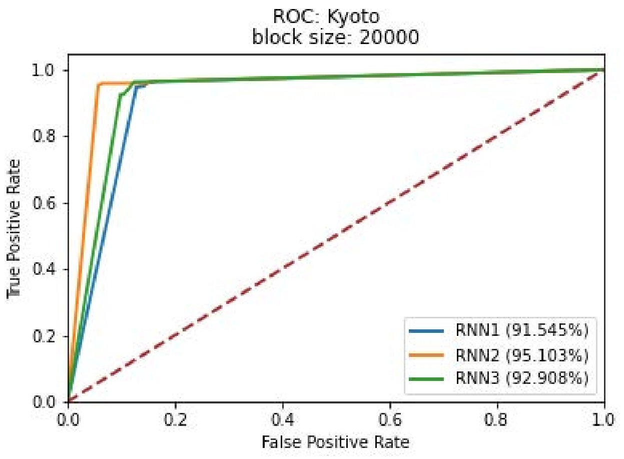


FIGURE 9. Performance evaluation with Kyoto 2006 dataset ($\beta = 20K$).

In Kyoto 2006 dataset, the RNN2 case performs relatively the best for the 20K block size, whereas the RNN1 case performs relatively the best for the 40K block size. Under the 20K block size, RNN2 showed 95.103% True Positive Rate, whereas RNN1 and RNN3 showed 91.545% and 92.908% True Positive Rate respectively.

In CIDDS dataset, the difference among the three different cases is minuscule. In terms of the block size, the 20K block size performs better than the 40K block size. True Positive Rates are at least 99.900% in both block sizes, but the one from 20K block size is relatively higher.

These experiments demonstrated that the data size for model building does not significantly impact the system performance. The sophisticated system settings in the algorithm is the most important to generate the best model in real-time with a given small amount of data. Note that the experimental results in this section are related to the best-effort approach. But when we used the adaptive feature engineering approach also showed similar results with the best-effort approach.

E. ATTACK IMPACT

The proposed system keeps updating the generated model depending on the system parameters according to the

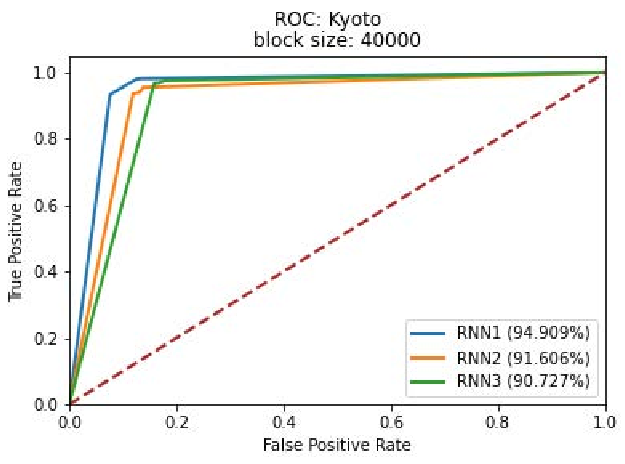


FIGURE 10. Performance evaluation with Kyoto 2006 dataset ($\beta = 40K$).

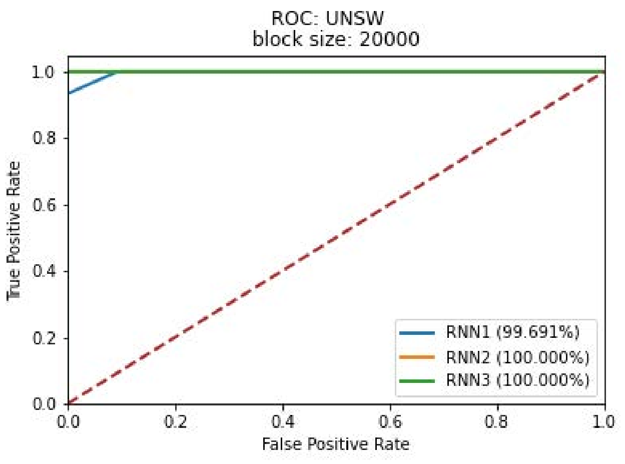


FIGURE 11. Performance evaluation with UNSW-NB15 dataset ($\beta = 20K$).

detection threshold. Attackers can intentionally generate obfuscated traffic to prevent our system from updating the generated model or to disturb our updated process for high false alerts. Thus, we made an experiment of our system under attacks to evaluate the robustness in which the proposed system can securely upgrade the current model with the new model under attacks. For this purpose, we utilized only NSL-KDD 99 and Kyoto 2006 out of the four datasets since we can match their feature sets between target data and attack data. The target data is actual dataset that the proposed system needs to generate the attack model for intrusion detection. The attack data is an obfuscated traffic that attackers intentionally generate to interrupt our updated process for the model generation. The target data is the NSL-KDD 99 dataset, and the Kyoto 2006 dataset is used as attack data. We set up our system parameters: 50,000 traces for the window size (ϖ), 20,000 traces for the block size (β), and 70% for the detection threshold (δ).

Figure 15 showed the result of our experiments under attack. Our baseline shows 98.086% True Positive Rate and 2.376% False Positive Rate with the NSL-KDD 99 dataset. We used the following feature sets for this

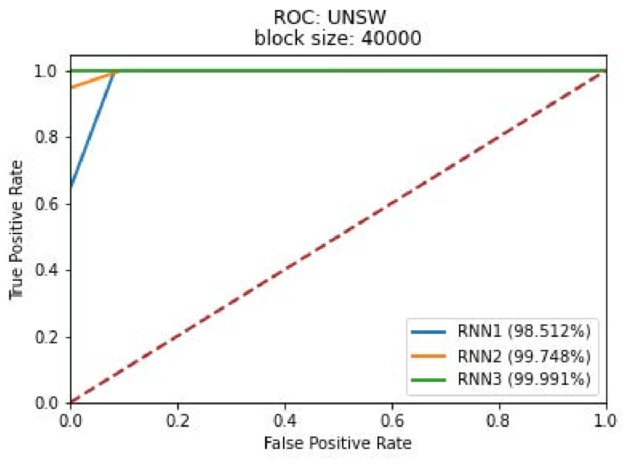


FIGURE 12. Performance evaluation with UNSW-NB15 dataset ($\beta = 40K$).

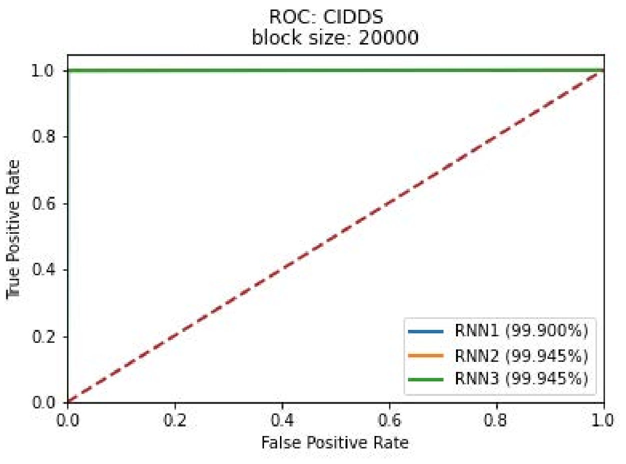


FIGURE 13. Performance evaluation with CIDDS dataset ($\beta = 20K$).

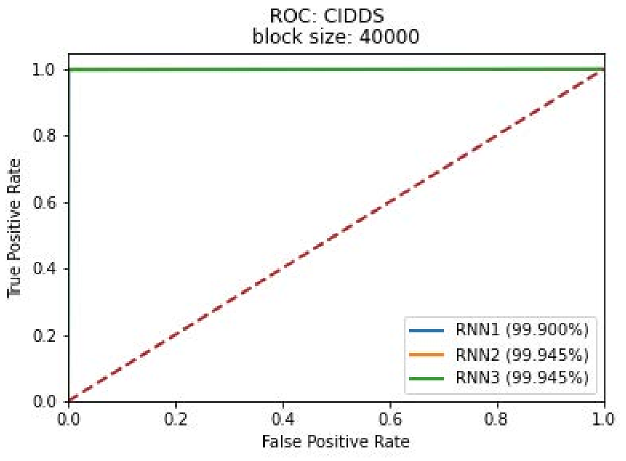


FIGURE 14. Performance evaluation with CIDDS dataset ($\beta = 40K$).

experiment: src_bytes, dst_bytes, difficulty_level, flag, and same_srv_rate.

Under the 20% attack rate that attackers randomly generate 10,000 obfuscated traces, the proposed system showed 96.855% True Positive Rate and 0.992% False Positive Rate.

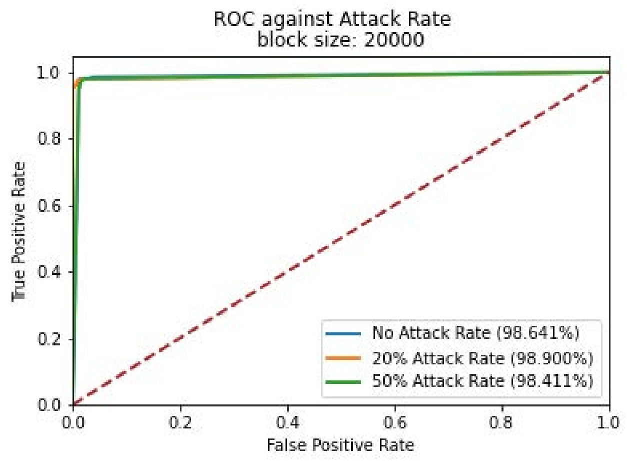


FIGURE 15. **Performance evaluation with the proposed system under various attack Rate ($\varpi = 50K$ and $\beta = 20K$).**

Under the 50% attack rate that attackers randomly generate 15,000 obfuscated traces, our system resulted in 96.584% True Positive Rate and 1.618% False Positive Rate. Figure 15 demonstrated that our proposed system achieved around 98% accuracy when attackers can generate less than 50% attack traffic to interrupt our updated process. Due to the space limitation, we drop out other attack cases for different system parameter settings. Therefore, our system will be used for future model generation. It can securely build a model in real time since attackers cannot generate the exact traffic pattern at a specific time.

F. DATA PROCESSING TIME

The proposed system has been developed by combining RNN with the Random Forest algorithm by setting various system parameters. We evaluated the entire processing time to generate an attack model from the data processing to the model building for two methods, respectively. We evaluated the system time for RNN and Random Forest with 500MB and 1GB datasets for each. As a result, the Random Forest algorithm spent 3.39570 minutes for 1GB and 1.71211 minutes for 500MB, but the RNN method required 43.32197 minutes for 1GB and 24.61870 minutes for 500MB. Therefore, because the deep learning techniques require more time than the machine learning techniques, it is not appropriate to utilize the RNN method in a real environment. Thus, we first utilized the Random Forest algorithm to select the best subset used for input for the next RNN technique as shown in Figure 1. The best subset included only high-quality data after deleting ambiguous data. Since the window size used for these experiments is much smaller than 500MB, the system time of the proposed system was less than 2-3 minutes. In detail, the 50K window size was 2MB, the 100K window size was 4M, and the 150K window size was 6.1MB. With the high specification of the experiment machine than ours, we expect that the system time to generate a model will significantly drop to several milliseconds.

V. DISCUSSION

This paper first proposed a real-time NIDS based on the combination of RNN and Random Forest with a reasonable data size. The goal of the proposed system continues improving the generated models by reflecting network dynamics in real time while considering the system parameters and feature sets. This section discusses the advantages and disadvantages of the proposed system with future work.

A. REAL-TIME MODEL BUILDING

To build a model in real time and to achieve the highest accuracy, the proposed system utilizes the machine learning technique first to reduce the data processing time and then applies the deep learning technique to generate an accurate attack model based on the well-classified selected data. As we discussed in the evaluation section, most deep learning techniques require a lot of processing and model building time while providing more advantages than other methods, such as automation, scalability, and effectiveness. With no previous knowledge, most deep learning methods automatically create an attack model through multiple layer processing with a large data size (i.e. more than 1TB). However, such nice features cannot be useful in a real network environment since network behavior is dynamically changing over time. The pre-built model cannot continuously monitor ever-changing network behavior. In addition, it is not practical to build up the attack model with such large data sizes due to time issues. Thus, to build or to update an attack model in real time, the system must create an accurate model with small high-quality data. To achieve this goal, the combination of the multi-classifier and deep learning solved two important issues: data classification and intelligent attack model generation in real-time.

B. TRAINING DATA SIZE

The training data size is important to build an accurate attack model. However, selecting high-quality right data is the most significant task before the model building. To build a real-time NIDS, the proposed system established three important system parameters: a window size (ϖ), a block size (β), and a random time (Δt). Based on these system parameters, the proposed system collects and selects training data in real time. When we consider the current network capacity (5G or 6G), the proposed system can collect enough data size within several microseconds. Since a 10 Gbps gigabit network can transmit 1.25 gigabytes per second, the proposed system easily collects 50K to 100K traces (2M to 10MB) in real-time within less than one millisecond as we discussed in the evaluation section. And then, the Random Forest algorithm performs data classification to identify ambiguous data that reduce system performance for high accuracy. Through the experiments, this paper recommends that the window size (ϖ) would be from 50K to 100K. This paper showed that the largest data size that is more than 100K did not provide the highest accuracy through our experiments.

C. SYSTEM VULNERABILITY

As simulated in Section IV, attackers can intentionally inject their fake traffic into the network to prevent our system from updating the attack model correctly. However, since the proposed system collects and selects real-time data at a given random time (Δt), attackers cannot easily obfuscate current target traffic patterns. In addition, the window size (ϖ) and the block size (β) along with used feature sets are unknown. To compromise the proposed system, attackers must keep performing a brute force attack to inject fake data into the system. Furthermore, since the Random Forrest algorithm as a multi-classifier detects outliers or ambiguous data from the collected data, attackers cannot easily defeat the proposed system with a simple effort and time. However, as attack strategies continue to evolve, the proposed system needs to keep fortifying itself from other possible attack cases.

In future work, we will develop a more advanced real-time network intrusion detection system by challenging the above-mentioned issues while developing various attack methods to improve robustness.

VI. CONCLUSION

In this paper, we proposed a flexible and robust NIDS by using RNN while updating the built-in attack models in real-time by considering current network behavior and performance. The proposed system utilizes RNN with a multi-classifier in order to randomly select new data sets depending on system parameters in the system. Our system also has random features to prevent attackers from obfuscating current traffic to interrupt the model generation in real-time.

We have found that the combination of machine learning and deep learning serves two ends: high accuracy in modeling and real-time detection. Random Forest enables us to achieve real-time detection due to high performance for data classification, while RNN utilizes its innate sequential data parsing to increase high accuracy. Based on the hyperparameter tuning, we have found that a combination of Sigmoid activation, 0.15 dropout rate, and 0.1 or 0.05 learning rate leads to the highest metric. A block size matters; 20K performs better than 40K in terms of accuracy. On the other hand, a window size of 50K or 100K does provide a better performance in the proposed system. We demonstrated that our proposed system achieved around 98% accuracy when attackers can generate less than 50% of attack traffic to interrupt our updated process.

By utilizing both multi-classifier and deep learning with the random system parameters, our proposed framework can provide significant contributions in the direction of the real-time network intrusion detection systems.

REFERENCES

[1] A. Abubakar and B. Pranggono, "Machine learning based intrusion detection system for software defined networks," in *Proc. 7th Int. Conf. Emerg. Secur. Technol. (EST)*, Sep. 2017, pp. 138–143.

[2] S. K. Dey and M. M. Rahman, "Flow based anomaly detection in software defined networking: A deep learning approach with feature selection method," in *Proc. 4th Int. Conf. Electr. Eng. Inf. Commun. Technol. (iCEEiCT)*, Dhaka, Bangladesh, Sep. 2018, pp. 630–635.

[3] C. Song, W. Fan, S.-Y. Chang, and Y. Park, "Reconstructing classification to enhance machine-learning based network intrusion detection by embracing ambiguity," in *Proc. Silicon Valley Cybersecurity Conf.*, 2020, pp. 169–187.

[4] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, "Machine-learning based threat-aware system in software defined networks," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2017, pp. 1–9, doi: 10.1109/ICCCN.2017.8038436.

[5] F. Chollet, *Deep Learning with Python*. New York, NY, USA: Manning, 2017, p. 198.

[6] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of Statistical Learning*. Berlin, Germany: Springer, 2009.

[7] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019, doi: 10.1109/ACCESS.2019.2912200.

[8] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[9] G. Williams, R. Baxter, H. He, S. Hawkins, and L. Gu, "A comparative study of RNN for outlier detection in data mining," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2002, pp. 709–712, doi: 10.1109/ICDM.2002.1184035.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[11] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Inf. Sci.*, vol. 378, pp. 484–497, Feb. 2017.

[12] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[13] J. Song, H. Takakura, and Y. Okabe, "Description of Kyoto university benchmark data," Academic Center Comput. Media Studies (ACCMS), Kyoto Univ., Kyoto, Japan, Tech. Rep., Dec. 2015.

[14] A. Verma and V. Ranga, "Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning," *Proc. Comput. Sci.*, vol. 125, pp. 709–716, Jan. 2017, doi: 10.1016/j.procs.2017.12.091.

[15] T. Mehmood and H. B. Md Rais, "Machine learning algorithms in context of intrusion detection," in *Proc. 3rd Int. Conf. Comput. Inf. Sci. (ICCOINS)*, Aug. 2016, pp. 369–373, doi: 10.1109/ICCOINS.2016.7783243.

[16] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, *arXiv:1901.03407*.

[17] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Appl. Sci.*, vol. 9, no. 20, p. 4396, Oct. 2019, doi: 10.3390/app9204396.

[18] M. Mayhew, M. Atighetchi, A. Adler, and R. Greenstadt, "Use of machine learning in big data analytics for insider threat detection," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Canberra, ACT, Australia, Oct. 2015, pp. 915–922.

[19] L. Hu, T. Li, N. Xie, and J. Hu, "False positive elimination in intrusion detection based on clustering," in *Proc. 12th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, Zhangjiajie, China, Aug. 2015, pp. 519–523.

[20] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Secur. Commun. Netw.*, vol. 2018, pp. 1–9, Jul. 2018, doi: 10.1155/2018/4943509.

[21] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "Deep-full-range: A deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45182–45190, 2019, doi: 10.1109/ACCESS.2019.2908225.

[22] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Secur. Commun. Netw.*, vol. 2017, pp. 1–10, Nov. 2017, doi: 10.1155/2017/4184196.

[23] M. Rigaki and S. Garcia, "Bringing a GAN to a knife-fight: Adapting malware communication to avoid detection," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, San Francisco, CA, USA, May 2018, pp. 70–75.

[24] K. Goeschel, "Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis," in *Proc. SoutheastCon*, Norfolk, VA, USA, Mar. 2016, pp. 1–6.

[25] P. Kuttranont, K. Boonprakob, C. Phaudphut, S. Permpol, P. Aimtongkhamand, U. KoKaew, B. Waikham, and C. So-In, "Parallel KNN and neighborhood classification implementations on GPU for network intrusion detection," *J. Telecommun., Electron. Comput. Eng.*, vol. 9, pp. 29–33, Jun. 2017.

[26] K. Peng, V. C. M. Leung, and Q. Huang, "Clustering approach based on mini batch Kmeans for intrusion detection system over big data," *IEEE Access*, vol. 6, pp. 11897–11906, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8304564

[27] S. Potluri, S. Ahmed, and C. Diedrich, "Convolutional neural networks for multi-class intrusion detection system," in *Mining Intelligence and Knowledge Exploration*. Cham, Switzerland: Springer, 2018, pp. 225–238.

[28] B. Zhang, Y. Yu, and J. Li, "Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Kansas City, MO, USA, May 2018, pp. 1–6.

[29] H. Zhang, X. Yu, P. Ren, C. Luo, and G. Min, "Deep adversarial learning in intrusion detection: A data augmentation enhanced framework," 2019, *arXiv:1901.07949*.

[30] S. Teng, N. Wu, H. Zhu, L. Teng, and W. Zhang, "SVM-DT-based adaptive and collaborative intrusion detection," *IEEE/CAA J. Automat. Sinica*, vol. 5, no. 1, pp. 108–118, Jan. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8232594

[31] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, Oct. 2016, doi: 10.3390/s16101701.

[32] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, and H. Janicke, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *Proc. 15th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, Santorini Island, Greece, May 2019, pp. 228–233.

[33] F. A. A. Alseiari and Z. Aung, "Real-time anomaly-based distributed intrusion detection systems for advanced metering infrastructure utilizing stream data mining," in *Proc. Int. Conf. Smart Grid Clean Energy Technol. (ICSGCE)*, Offenburg, Germany, Oct. 2015, pp. 148–153.

[34] X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS attack via deep learning," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Hong Kong, May 2017, pp. 1–8.

[35] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," 2018, *arXiv:1803.10769*.

[36] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8171733

[37] W. Meng, W. Li, and L.-F. Kwok, "Design of intelligent KNN-based alarm filter using knowledge-based alert verification in intrusion detection," *Secur. Commun. Netw.*, vol. 8, no. 18, pp. 3883–3895, Dec. 2015, doi: 10.1002/sec.1307.

[38] S. McElwee, J. Heaton, J. Fraley, and J. Cannady, "Deep learning for prioritizing and responding to intrusion detection alerts," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Baltimore, MD, USA, Oct. 2017, pp. 1–5.

[39] N. N. Tran, R. Sarker, and J. Hu, "An approach for host-based intrusion detection system design using convolutional neural network," in *Proc. Int. Conf. Mobile Netw. Manage.* Berlin, Germany: Springer, Sep. 2017, pp. 116–126.

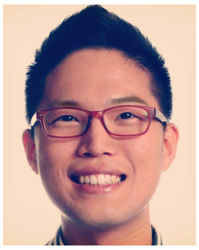
[40] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *Proc. Workshops 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, Feb. 2017, pp. 1–9.

[41] A. Bohara, U. Thakore, and W. H. Sanders, "Intrusion detection in enterprise systems by combining and clustering diverse monitor data," in *Proc. Symp. Bootcamp Sci. Secur.*, Pittsburgh, PA, USA, Apr. 2016, pp. 7–16.

[42] S. O. Uwagbole, W. J. Buchanan, and L. Fan, "Applied machine learning predictive analytics to SQL injection," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, May 2017, pp. 1087–1090.

[43] A. M. Vartouni, S. S. Kashi, and M. Teshnehlab, "An anomaly detection method to detect Web attacks using stacked auto-encoder," in *Proc. 6th Iranian Joint Congr. Fuzzy Intell. Syst. (CFIS)*, Kerman, Iran, Feb. 2018, pp. 131–134.

[44] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.



KICHO YU received the B.S. degree in actuarial science and statistics from Purdue University, and the M.S. degree in data analytics from the University of San Francisco. He is currently pursuing the M.S. degree in computer science with Northeastern University. He is also a Research Assistant in cybersecurity at Northeastern University. He is a Vice President and the Program Director of the Coding School at Bay Area K Group—a non-profit organization—where he teaches coding to the second and third generation immigrant teenagers. He taught machine learning and mentored students at the Silicon Valley Cybersecurity Institute (SVCSI). Before he joined academia, he worked as a Data Analyst and a Business Intelligence Engineer in the San Francisco Bay Areas in different verticals: from cybersecurity, insurance, transportation, to healthcare. He worked at various companies, such as Lyft, Allstate (SquareTrade), AnchorFree (Aura), Nuna Health, and Acumen LLC. He is a Korean Army Special Forces Veteran.

KHANH NGUYEN is currently pursuing the degree in software engineering with San José State University. She is also working as a Research Assistant and as a Web Developer with the Silicon Valley Cybersecurity Institute (SVCSI). Her research interests include machine learning and the cybersecurity.

YOUNGHEE PARK (Member, IEEE) received the Ph.D. degree in computer science from North Carolina State University, in 2010. She worked at the National Security Research Institute, ETRI, South Korea, in 2003. She is currently an Associate Professor of computer engineering at San José State University. Before joining this department, in August 2013, she was a Postdoctoral Researcher at the University of Illinois at Urbana–Champaign, in 2013, and at Columbia University in New York City, in 2011. She is the President and the Founder of the Silicon Valley Cybersecurity Institute (SVCSI), a public non-profit organization. She was a Visiting Professor at IBM Almaden Research, from July 2019 to August 2020, and the Vice-Chair of the IEEE Computer Society, Santa Clara Valley Chapter, for one year in 2021. Her research interests include networks and system security. Her research has been funded by many industries and the National Science Foundation (NSF). She obtained an Award of Excellence as a Distinguished Faculty Mentor for the SJSU Student Research Competition in 2017. She was selected for the Kordestani Endowed Chair in the College of Engineering in 2016 and 2017 as a Distinguished Research Professor. She also received the Faculty Award for Excellence in Scholarship in the College of Engineering, SJSU, in 2018.