IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SUBMISSION.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Cyber Threat Detection based on Artificial Neural Networks using Event Profiles

JONGHOON LEE[1,2], JONGHYUN KIM[1], IKKYUN KIM[1], and KIJUN HAN[2]

[1]Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, South Korea,
[2]School of Computer Science and Engineering, Kyungpook National University, Daegu 41566, South Korea,

Corresponding author: Kijun Han(e-mail: kjhan@knu.ac.kr).

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP), a grant funded by Korea government (Ministry of Science and ICT) (no. 2016-0-00078, Cloud Based Security Intelligence Technology Development for the Customized Security Service Provisioning).

ABSTRACT One of the major challenges in cybersecurity is the provision of an automated and effective cyber-threats detection technique. In this paper, we present an AI technique for cyber-threats detection, based on artificial neural networks. The proposed technique converts multitude of collected security events to individual event profiles and use a deep learning-based detection method for enhanced cyber-threat detection. For this work, we developed an AI-SIEM system based on a combination of event profiling for data preprocessing and different artificial neural network methods, including FCNN, CNN, and LSTM. The system focuses on discriminating between true positive and false positive alerts, thus helping security analysts to rapidly respond to cyber threats. All experiments in this study are performed by authors using two benchmark datasets (NSLKDD and CICIDS2017) and two datasets collected in the real world. To evaluate the performance comparison with existing methods, we conducted experiments using the five conventional machine-learning methods (SVM, k-NN, RF, NB, and DT). Consequently, the experimental results of this study ensure that our proposed methods are capable of being employed as learning-based models for network intrusion-detection, and show that although it is employed in the real world, the performance outperforms the conventional machine-learning methods.

INDEX TERMS Cyber security, intrusion detection, network security, artificial intelligence, deep neural networks.

I. INTRODUCTION

With the emergence of artificial intelligence (AI) techniques, learning-based approaches for detecting cyber attacks, have become further improved, and they have achieved significant results in many studies. However, owing to constantly evolving cyber attacks, it is still highly challenging to protect IT systems against threats and malicious behaviors in networks. Because of various network intrusions and malicious activities, effective defenses and security considerations were given high priority for finding reliable solutions [1], [2], [3], [4].

Traditionally, there are two primary systems for detecting cyber-threats and network intrusions. An intrusion prevention system (IPS) is installed in the enterprise network, and can examine the network protocols and flows with signature-based methods primarily. It generates appropriate intrusion alerts, called the security events, and reports the generating alerts to another system, such as SIEM. The security information and event management (SIEM) has been focusing on collecting and managing the alerts of IPSs. The SIEM is the most common and dependable solution among various security operations solutions to analyze the collected security events and logs [5]. Moreover, security analysts make an effort to investigate suspicious alerts by policies and threshold, and to discover malicious behavior by analyzing correlations among events, using knowledge related to attacks.

Nevertheless, it is still difficult to recognize and detect intrusions against intelligent network attacks owing to their high false alerts and the huge amount of security data [6], [7]. Hence, the most recent studies in the field of intrusion detection have given increased focus to machine learning and artificial intelligence techniques for detecting attacks. Advancement in AI fields can facilitate the investigation of

network intrusions by security analysts in a timely and automated manner. These learning-based approaches require to learn the attack model from historical threat data and use the trained models to detect intrusions for unknown cyber threats [8], [9].

A learning-based method geared toward determining whether an attack occurred in a large amount of data can be useful to analysts who need to instantly analyze numerous events. According to [10], information security solutions generally fall into two categories: analyst-driven and machine learning-driven solutions. Analyst-driven solutions rely on rules determined by security experts called analysts. Meanwhile, machine learning-driven solutions used to detect rare or anomalous patterns can improve detection of new cyber threats [10]. Nevertheless, while learning-based approaches are useful in detecting cyber attacks in systems and networks, we observed that existing learning-based approaches have four main limitations.

First, learning-based detection methods require labeled data, which enable the training of the model and evaluation of generated learning models. Furthermore, it is not straightforward to obtain such labeled data at a scale that allow accurate training of a model. Despite the need for labeled data, many commercial SIEM solutions do not maintain labeled data that can be applied to supervised learning models [10].

Second, most of the learning features that are theoretically used in each study are not generalized features in the real world, because they are not contained in common network security systems [3]. Hence, it makes difficult to utilize to practical cases. Recent efforts on intrusion detection research have considered an automation approach with deep learning technologies, and performance has been evaluated using well-known datasets like NSLKDD [11], CICIDS2017 [12], and Kyoto-Honeypot [13]. However, many previous studies used benchmark dataset, which, though accurate, are not generalizable to the real world because of the insufficient features. To overcome these limitations, an employed learning model requires to evaluate with datasets that are collected in the real world.

Third, using an anomaly-based method to detect network intrusion can help detect unknown cyber threats; whereas it can also cause a high false alert rate [6]. Triggering many false positive alerts is extremely costly and requires a substantially large amount of effort from personnel to investigate them.

Fourth, some hackers can deliberately cover their malicious activities by slowly changing their behavior patterns [10], [14]. Even when appropriate learning-based models are possible, attackers constantly change their behaviors, making the detection models unsuitable. Moreover, almost all security systems have been focused on analyzing short-term network security events. To defend consistently evolving attacks, we assume that over long-term periods, analyzing the security event history associated with the generation of events can be one way of detecting the malicious behavior of cyber attacks.

These challenges form the primary motivation for this work. To address these challenges, we present an AI-SIEM system which is able to discriminate between true alerts and false alerts based on deep learning techniques. Our proposed system can help security analysts rapidly to respond cyber threats, dispersed across a large amount of security events.

For this, the proposed the AI-SIEM system particularly includes an event pattern extraction method by aggregating together events with a concurrency feature and correlating between event sets in collected data. Our event profiles have the potential to provide concise input data for various deep neural networks. Moreover, it enables the analyst to handle all the data promptly and efficiently by comparison with long-term history data.

The main contributions of our work can be summarized as follows:

- Our proposed system aims at converting a large amount of security events to individual event profiles for processing very large scale data. We developed a generalizable security event analysis method by learning normal and threat patterns from a large amount of collected data, considering the frequency of their occurrence. In this study, we specially propose the method to characterize the data sets using the basepoints in data preprocessing step. This method can significantly reduce the dimensionality space, which is often the main challenge associated with traditional data mining techniques in log analysis.
- Our event profiling method for applying artificial intelligence techniques, unlike typical sequence-based pattern approaches, provides featured input data to employ various deep-learning techniques. Hence, because our technique is able to facilitate improved classification for true alerts when compared with conventional machine-learning methods, it can remarkably reduce the number of alerts practically provided to the analysts.
- For the applicability, we evaluate our system with real IPS security events from a real security operations center (SOC) and validate its effectiveness through performance metrics, such as the accuracy, true positive rate (TPR), false positive rate (FPR) and the F-measure. Moreover, to evaluate the performance comparison with existing methods, we conducted experiments using the five conventional machine-learning methods (SVM, k-NN, RF, NB and DT). And we also perform an evaluation by applying our method to two benchmark datasets (i.e., NSLKDD, CICIDS2017), which are most commonly used in the field of network intrusion detection research.

In this study, to decompose a large amount of collecting events into individual event occurrence profiles, we apply the TF-IDF mechanism. We also generate the event profiles by computing the similarity value among each TF-IDF event sets and appointed basepoints. The generated event profiles are fed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953095, IEEE Access

into the input-layer of the FCNN, CNN, and LSTM models, which are executed in AI-SIEM. Consequently, using two well-known benchmark datasets and two real datasets collected from operating IPS, we aim to show the applicability of our system for defending IT systems against the cyber threats.

For evaluation, we are aware of the limitation of NSLKDD and CICIDS2017 datasets, but they remain widely used benchmarks for comparing machine-learning methodologies. Hence, we also conduct a performance comparison with existing methods using the real datasets and additional two benchmark datasets. Above all, machine-learning approaches obtained a good performance using benchmark datasets, also need to achieve satisfactory performance for the real data.

The remainder of this paper is structured as follows. In Section II, we introduce the background information for the proposed system. Section III provides existing works on learning-based intrusion or attack detection. In Section IV, we describe the overview for our proposed system and data labeling. In section V, we specify the methodology used in this study in more detail. Section VI provides the implementation of the FCNN, CNN, and LSTM models for this study. Section VII introduces datasets for experiments. Section VIII presents the detailed evaluation results of experiments and comparison with other methods. Finally, the conclusion and future work discussed in Section IX

II. PRELIMINARIES

In this section, we shortly discuss the background information for our study. We start by describing the overview of the IDS/IPS and the SIEM, and introduce the deep learning techniques. Finally, we describe our big data platform for the proposed AI-SIEM system.

A. IDS / IPS and SIEM

1) IDS / IPS

An intrusion detection system (IDS) monitors the network activity and reports on observation of any security violations [6]. Unlike the IDS, an intrusion prevention system (IPS) can block a detected network connection by closing port or dropping the packets. An IPS has become an indispensable system for most types of organizations or industries owing to the wide growing nature of data and the internet. Nevertheless, intelligent network attacks still persist in today′s network, and there are limitations to detect and respond network intrusions by an IPS system [15]. This is because they mainly use less-capable signature-based detection, as opposed to anomaly detection methods. Meanwhile, speedy attacks are occurring more frequently with new intrusion methods [6], [16]. Most of all, the majority of IPS solutions have a high false positive rate and are limited in detecting any unknown or new attacks. In addition, in [14], the authors presented six limitations for an IPS such as the challenges of volume, accuracy, diversity, dynamics, low-frequency attacks, and adaptability. These limitations lead to seriously restrict precise decision by an SOC security analyst.

2) SIEM

A SIEM has been considered an important component of enterprise networks and security infrastructures, with a focus on enterprise information technology (IT) security, which provides an overall view of the security management. In general, SIEM collects relevant data produced in an organization from various sources, making it possible to detect cyber threats by matching patterns [17], [18], [19]. The SIEM system allows the consolidation and comprehensive evaluation of security alerts and logs collected from network security systems (e.g., firewall and IDS / IPS). Particularly with analyzing IDS/IPS alerts (security events) in SIEM, the analyst make an effort to find cyber attacks using pre-defined security policies and threshold. Moreover, to discover consolidated malicious behavior, they carry out analyzing correlations between security events and relevant situations based on already known patterns of cyber threats.

Security events are continually generated from many types of network security systems (e.g., IPS and FW); thus, they are heterogeneous with an extremely diverse distribution. This brings challenges to discriminate true positive alerts from false ones in a traditional policy-based threat detection system. Moreover, practice shows that this method of analyzing is extremely complex, high costly and only operable with large personnel effort [18].

For cyber-threat detection, the SIEM analysts spend an immense amount of effort and time to differentiate between true security alerts and false security alerts in collected events. Hence, in recent years, to address this challenge, one of the main focuses within the development of SIEM has been the application of machine-learning and artificial-intelligence (AI)-learning techniques, which is referred to here as AI-based SIEM. Although the application of these techniques has offered improvement in reducing human labor, there are still several challenges for an AI-based SIEM. As mentioned above, there are major limitations such as (1) the comparatively high level of analyst interaction required, (2) lack of labeled data, and (3) constantly evolving attacks [10], [14].

B. DEEP LEARNING TECHNIQUES

In recent years, the deep learning technique has been greatly advanced in many areas, and it is ongoing in many industries beyond an area of machine learning that applies neurons as mathematical structures similar to human neural network. The most widely used deep neural network are convolutional model and recurrent model.

CNNs are generally effective to learn the spatial features of data such as image processing, and RNNs are the more suitable method that can learn using time-continuously differentiable features of data. CNNs are architectures especially designed to deal with spatial data. Because of the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953095, IEEE Access

awareness of the partially specific feature of the input, specific local characteristic, and shared parameter schemes, CNNs are employed in many fields [20], [21], [22]. CNNs have already yielded remarkable outcomes in many fields such as image classification [23], biomedical text analysis [24], and malware classification [3], [25], [26], [27], [28], [29]. For network intrusion detection, many studies showed the feasibility of CNN for the identification of malicious events, network flow and connection in the network [30], [31].

Recurrent structures are capable of learning the sequence information in the data. The well-known recurrent structures are RNN and LSTM [32], [33]. LSTM has a special recurrent architecture designed to advance the storage ability, compared to RNNs. This is mainly because RNN is able to store past input information for short time, that degrades its ability to model a long-term structure for the input sequence [34]. Hence, LSTM networks have an additional component called the forget gate. Because LSTM can effectively perform to learn long sequence data, it also has enabled successfully empirical results in areas such as speech recognition and machine translation [3], [10].

C. BIG DATA PLATFORM

Typically, a big data platform is used to collect data on security events from IPS and maintain security logs over long-term periods. The big data platform can also be specialized in analyzing data and quickly recognizing cyber threats [35], [36]. This is because historical data collected over long-term periods in the platform can help investigate and respond to cyber threats. For this, we have developed the scalable big data platform based on distributed computing technologies, particularly for collecting, processing, storing, correlating, and analyzing the security event logs.

Figure 1 shows the system architecture of our big data platform. The platform mainly consists of a data collection system, data processing system, data analysis and data storage system to analyze cyber-threat information using long-term security data. Using the techniques for large-scaled data processing, this platform is capable of continually collecting the numerous streamed security events and processing the data in real-time [37]. Based on the big data platform, our proposed methods can be coupled with AI-based SIEM. In this work, by adopting AI technique to the platform, true alerts can be better differentiated from false alerts in the real world.

III. RELATED WORKS

In this section, we discuss previous studies for deep learning-based intrusion detection and real security event analysis research. In recent years, many studies in cybersecurity focus on AI-based intrusion detection, and different AI and machine learning-based techniques have been proposed to improve the ability of cyber threat detection [1], [2], [3], [15], [38], [39]. Although these studies have achieved significant result using AI and machine learning-based techniques, they are still limited to specific test datasets such as NSLKDD. Other research studies however, have used security events and logs collected from the real world [8], [10], [40], [41], [42]. These studies are closer to our study for addressing the above-mentioned challenges. Especially, Liao *et al.* [39], Du *et al.* [40], and K. Zhang *et al.* [42] have used the TF-IDF mechanism like our method.

A. DEEP LEARNING-BASED INTRUSION DETECTION

Naseer *et al.* [1] proposed, implemented and trained intrusion detection models using different deep neural network architectures including CNNs, Autoencoders, and RNNs. These models were trained on the NSLKDD training dataset and evaluated on both test datasets provided by NSLKDD. DCNN and LSTM models showed a performance of 85% and 89% accuracy, respectively, on test dataset.

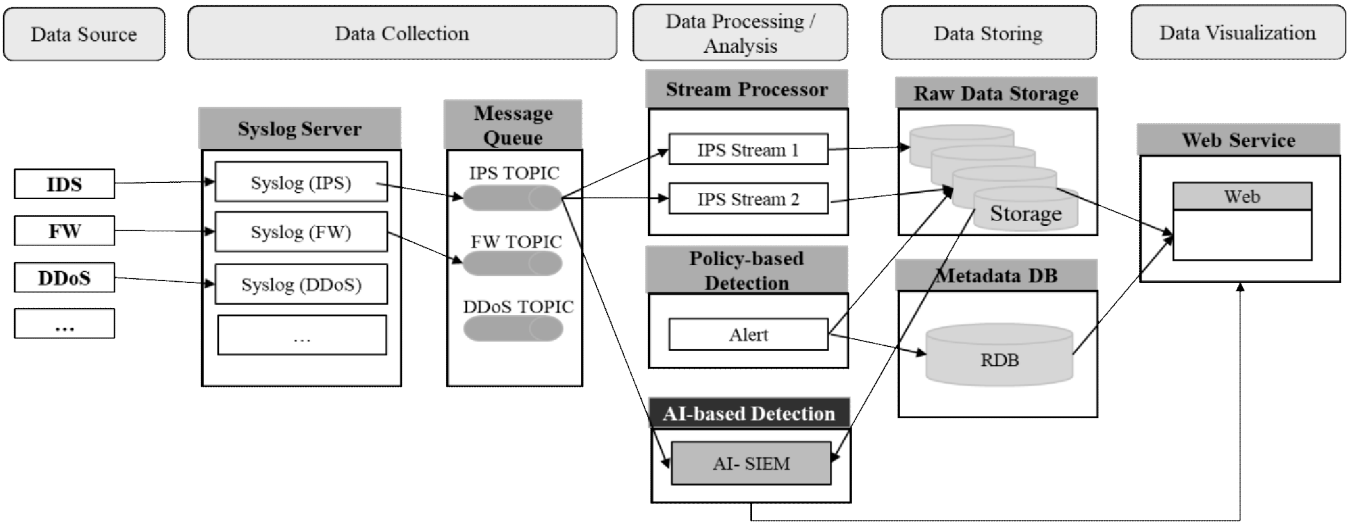

FIGURE 1. The architecture of our big data platform for AI-based SIEM

B. Zhang et al. [2] divided methods for network intrusion detection into two types: direct methods using single algorithm and combination method by combination of several methods. The author proposed a new detection model based on a directed acyclic graph (DAG) and a belief rule base (BRB). The results showed that compared with conventional detection models, the DAG-BRB combination model had a higher detection rate using KDD 99 dataset.

Wang et al. [3] proposed a hierarchical spatial and temporal features-based intrusion detection system (HAST-IDS) that automatically learns network traffic features. The main idea is that the spatial features of network traffic are first learned using deep CNNs and then learns the temporal features are learned LSTM networks. The experiments were conducted by DARPA and ISCX datasets.

Vinayakumar *et al.* [15] developed a hybrid intrusion detection system which has the capability to analyze the network and host-level activities. It employed distributed deep learning model with DNN for processing and analyzing very large scale data in real-time. The DNN model was selected by comprehensively evaluating their performance in comparison to classical machine learning classifiers on various benchmark IDS datasets such as NSLKDD and UNSW-NB15.

Khan *et al.* [38] propose a novel two-stage deep learning model, based on a stacked auto-encoder with a soft-max classifier, for efficient network intrusion detection. The authors conducted several experiments on two public datasets: the benchmark KDD99 and UNSW-NB15 datasets. This study achieved results, up to 99.9% for the KDD99 dataset and 89.1% for the UNSW-NB15 dataset.

Liao *et al.* [39] proposed a new algorithm based on the k-NN classifier method using TF-IDF for modeling program behavior in intrusion detection regarding system calls. In [29], with the k-NN classifier, the frequencies of system calls are used to describe the program behavior. For this, text categorization techniques, such as TF-IDF, are adopted to transform each system call data to a vector and measure the similarity between two program system call activities. Authors report that the TF-IDF-based k-NN classifier appears to be well applicable to the domain of intrusion detection in the field of malware detection.

B. REAL SECURITY EVENT ANALYSIS

Shen *et al.* [8] developed the system for predicting security events through deep learning, which is called Tiresias. Authors presented a system that leverages RNNs to predict future events on a machine, based on previous observations. It tested on a dataset of 3.4 billion security events collected from a commercial IPS, and showed that its approach is effective in predicting the next event that will occur on a machine with a precision of up to 0.93. In addition, the system also accomplished a high precision for a complex situation and maintained stable results.

Veeramachaneni *et al.* [10] developed end-to-end machine learning techniques that predict cyber attacks significantly better than existing systems by continuously incorporating input from human experts. The analyst directly labeled data with a ranked metric over several months, and these labeled data were provided to the supervised learning module to predict whether an attack would occur. This study showed that the technique, using six anomaly detection methods, can detect 85 percent attacks, which is roughly three times better than previous benchmarks, while also reducing the number of false positives by a factor of 5. The system was tested on 3.6 billion pieces of data known as “log lines,” which were generated by millions of users over a period of three months. Specially, the hybrid approaches of auto-encoders have been recently proposed for anomaly detection.

Du *et al.* [40] proposed DeepLog, a deep neural network model employing LSTM to train a system’s log patterns (e.g., log key patterns and corresponding parameter value patterns) from normal execution. This work uses the term frequency inverse document frequency (TF-IDF) vector to the log key and parameter value anomaly detection models for identifying abnormal log entries. The author showed that DeepLog outperformed existing log-based anomaly detection methods, achieving an F-measure of 96% in HDFS data and an F-measure of 98% in OpenStack data.

Oprea *et al.* [41] used belief propagation to detect early-stage enterprise infection from DNS logs. They proposed a new framework based on belief propagation inspired from graph theory. They demonstrated that the techniques perform well on two large datasets. The authors achieved high accuracy on two months of DNS logs. Moreover, they apply the algorithms to 38TB of web proxy logs collected at the border of a large enterprise. This framework used "hints" data that was manually provided by the SOC security analysts.

K. Zhang *et al.* [42] proposed a novel system that automatic-ally parses streamed console logs and detects early warning signals for IT system failure prediction. The system used an automation approach with text mining techniques, such as term frequency - inverse document frequency (TF-IDF) and it employed LSTM for deal with specific labeled data in the training process. The paper compared proposed technology with state-of-the-art machine learning approaches and showed the advantage and potentials of the system in prediction of complex IT failures.

The closest study to this paper is Tiresias [8]. Tiresias focused on anomaly detection for prediction of event in a noisy environment with a wide variety of events. However, in order to improve the accuracy for event prediction, Tiresias used the sequence-based approach with RNN for occurred security events. Whereas we adopt the concurrency-based approach with deep-learning to address the limitation of sequence-based method, which is detailed in the next Section.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953095, IEEE Access

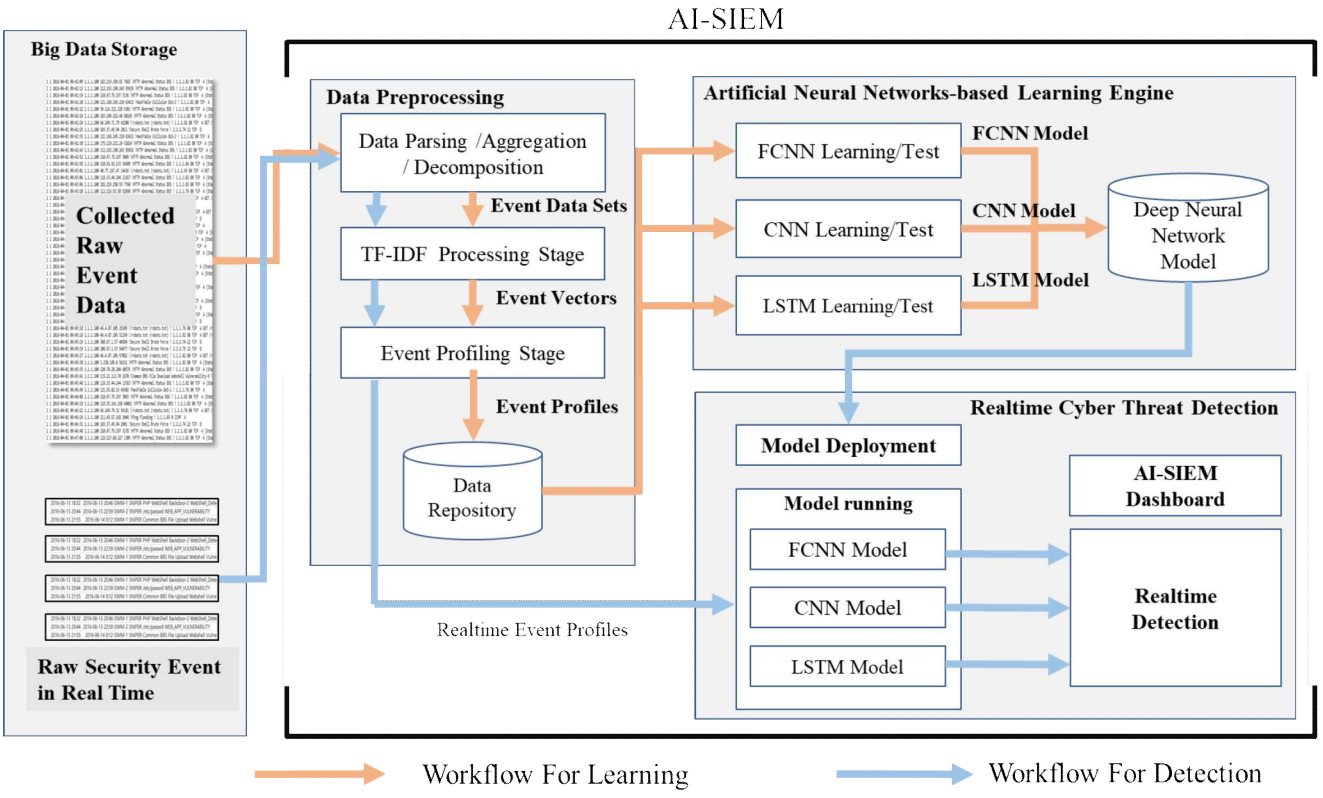

FIGURE 2. **The workflow and architecture for the developed AI-based SIEM system**

IV. SYSTEM OVERVIEW

This section describes the architecture of the proposed AI-SIEM system for artificial intelligence-based threat detection. The AI-SIEM system employs not only deep learning techniques but also data preprocessing mechanism that enables the handling of very large-scale network events. Specially, the main goal of the AI-SIEM is to automatically analyze network security events related to true alerts for detecting cyber-threats and execute multiple analysis engines. It also utilizes the processing capability of the several graphical processing unit (GPU) cores for faster and parallel analysis.

Figure 2 presents the workflow and architecture for the developed artificial intelligent (AI)-based SIEM system. The AI-SIEM system comprises three main phases: The data preprocessing, artificial neural networks-based learning engine, and real-time threat detection phase.

The first preprocessing phase in the system, termed event profiling, aims at providing concise inputs for various deep neural networks by transforming raw data. In the data preprocessing phase, data aggregation with parsing, data normalization stage using TF-IDF mechanism, and event profiling stage are consecutively performed in the AI-SIEM system. Each stage generates event data sets, event vectors, and event profiles, respectively, and the output is utilized in next each stage, as shown in Figure 2. This phase not only precedes the data learning stage but also precedes the conversion of raw security events to the deep-learning engine's input data when the system operates on detecting network intrusions in real time. The second AI-based learning engine employs three artificial neural networks for modeling. For the data learning stage, the preprocessed data are fed into the three artificial neural networks, and each ANN performs learning to find the most accurate model. Finally, in real-time threat detection, each ANN model mechanically classifies each security raw event using the trained model, and the dashboard shows the only recognized true alerts to security analysts for reducing false ones.

Each stage for data preprocessing is detailed in Section V, and second ANNs for data learning phase are described in Section VI.

A. DATA LABELING FOR LEARNING

In this subsection, we discuss the data labeling of security events for supervised learning. As mentioned above, to employ the supervised learning method, a labeled data is essential. For this, analysts should be able to label several months of data heuristically. In other words, analysts need to label the raw events as "Normal" or as "Threat," based on whether it belongs to a type of attack by analyzing correlations among raw security events. However, owing to a rapidly

growing number of security events and unknown cyber threats, the labeling of numerous data is time-consuming and costly. In addition, it is difficult to acquire the labeled security event dataset based on the action of SOC security experts in the real world.

By investigating occurred cyber attacks, most of detected attacks can be categorized as system hacking, denial of service, network attacks, scanning attacks, and suspicious authenti-cation activities. These attack types are determined by the SOC security analysts based on correlation among attack duration time, the number of attacker's IP, and importance of victim system.

In our study, to provide an available dataset for supervised learning, we had to carry out dataset labeling according to utilizing recorded information in the threat detection report list (e.g., attack start time, attack end time, and attacker's ip address information). The threat detection reports are made by the SOC analysts during raw data collecting periods. The labeling operation is automatically performed by the data labeling module in our system. First, the system extracts timestamps and network information from the threat detection report, for each recorded threat detection result. Next, the data labeling tool in the system, investigates correlation of extract-ed threat information on raw security event, with each threat using the big data platform. The security events that are correlated with IP address and time of each threat are labeled as "THREAT (Attack name)," and others are labeled as "NORMAL." The labeled result of our collected datasets is explained in Section VII.

V. METHODOLOGY

In this section, we describe an event profiling method for preprocessing. The method is composed of data aggregation and decomposition, TF-IDF normalization, and generating event profile. we first present an event set extraction method for the data preprocessing. Then, the event vectorization using TF-IDF for event profiles is described in detail. Finally, we present the event profiling method for inputs into three deep learning models. The proposed method was basically motivated by the observation that raw event data can be profiled by concurrent event sets. By combining each proposed method sequentially, the preprocessing for AI engine is operated as shown in Figure 2:

A. DATA AGGRERATION AND DECOMPOSITION

Finding a profiling method to represent a pattern in a large amount of data can help to summarize much information from the event data and utilize the inputs for deep learning.

To deal with a large amount of streaming event data in the real world, we needed a method to find one representative data set that identifies several events; thus, we generate the statistical event sets. The basic idea of our method is to extract the occurrence information regarding other events simultaneously generated with it. Once the raw events are collected in the big data platform, each event is mapped into one event set using the sliding window by predefined interval, which can belong to overlapping sets by configuration. In other words, the sliding window allows overlapping of one log over multiple profiles. In this, we apply a concurrency-based pattern instead of a sequence-based pattern [8], and the number of concurrency event name types in each event set is regarded as deterministic features for true-positive events. This is primarily because the ordering of events can change slightly based on unknown situations. For example, if there are two event sequences, a = 4→5→6→4→5→6, and sequence b = 4→5→6→5→4→6, the ordering of the two sequences is clearly different; however, the event occurrences of the two sequences are the same. However, in the real world, the sequence may be changed in IPS by system processes, resources, and network,; therefore we adopt the concurrency-based method that depends on co-occurrence information, which is not as tight as the sequence, but allows the calibration of the gap of changeable sequence.

Whenever the window slides at an interval of the pre-defined *time_interval*, each raw event data in the window is aggregated into several event sets $ES_{i,j}^{T=t}$ by source address S_i and destination address D_j.

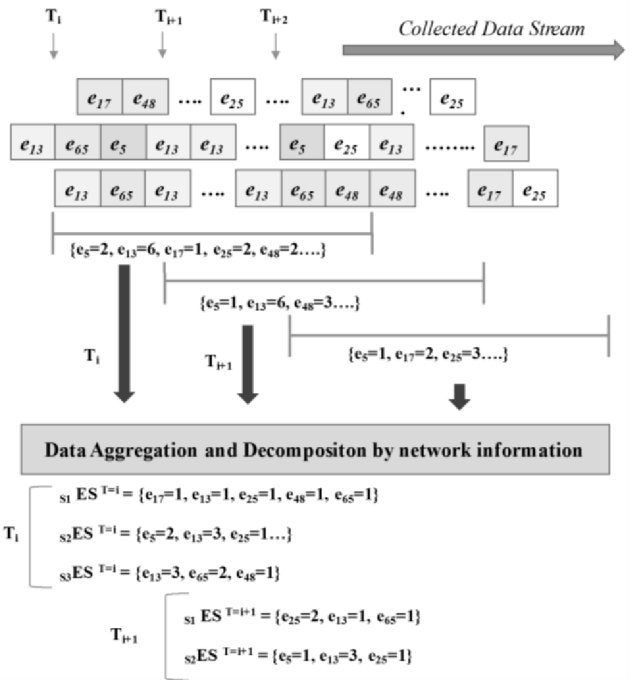

FIGURE 3. Data aggregation and data decomposition by source and destination address using sliding window.

Consequently, a number of event data sets are produced for one window time in our system. Figure 3 shows data aggregation and data decomposition by source and destination address using sliding window. For example, in case of first window T_i, occurred event set $\{ e_5 = 2, e_{13} = 6, e_{17} = 1, e_{25} = 2, e_{48} = 2, \cdots \}$ is decomposed to $_{S1}$ES = $\{e_{17} = 1, e_{13} = 1, e_{25} = 1 \ldots\}$, $_{S2}$ES = $\{e_5 = 2, e_{13} = 3, e_{25} = 1 \ldots\}$, and $_{S3}$ES=$\{e_{13} = 3, e_{65} = 2, e_{48} = 1 \ldots\}$ by connection

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953095, IEEE Access

unit. This operation is continuously performed on learning data.

B. TF-IDF DATA NORMALIZATION

In this subsection, event sets, which contain the frequency of unique event name such as event set ES_i, are transformed into a representation suitable for the learning algorithm and classifiers. For this, we use the vector space model which is the most commonly used document representation in the field of information retrieval.

We seek to adopt this technique to make an intrusion detection model. The occurrences of IPS events can be used to characterize the IPS pattern and transform each event set into a vector. Moreover, it is assumed that event sets belonging to the same concurrency will be nearby in vector space. Hence, as shown in Table 1, we substitute a different factor in threat detection for the concept of each factor in text categorization to apply the vector space model.

In the applied model, each event set is represented by a vector of occurred events. Note that m indicates the number of rows in the learning dataset, and n indicates the number of event name types. An m-by-n matrix E denotes the collection of event sets belonging to the learning dataset, where each entry represents the occurrence of an event in an event set, i.e., $E = (e_{ij})$, where e_{ij} is the weight of event j in event set i. There are several ways of determining weight e_{ij}.

Let tf_{ij} be the frequency of j th event in event set i, m the number of event sets in the entire dataset A, n is the number of unique event names in the entire dataset A, and n_j is the total number of times event j occurs in the entire collection.

Although there is simple Boolean weighting and frequency weighting, i.e., $e_{ij} = tf_{ij}$, the particular weighting approach is the so-called term frequency - inverse document frequency (TF-IDF) weighting as follows :

$$e_{ij} = tf_{ij} \times log\left(\frac{m}{n_j}\right) \quad (1)$$

TF-IDF is a statistical technique to index the term according to their importance, as it is based on vectors that represent the term frequency as well as term presence [43]. In this manner, the numerical value of a repeatedly occurring event exhibits a low weight, while the value of a very rarely occurring event will receive a high weight.

As a result of TF-IDF, matrix A is constructed, of which the columns length corresponds to the number of events M in the data collection, and the number of rows correspond to the number of event sets. Matrix A is composed of event vectors.

As mentioned in section III, Liao *et al.* [39] employed the TF-IDF for learning program behavior in malicious activities detection based on the frequencies of system calls invoked during a program execution time. Table 1 presents the substitution concept of TF-TDF for our AI-SIEM system.

Let a_{ij} denote the j th column TF-IDF value in the i th row of the dataset. To convert input data for deep learning with the above pre-processed dataset to corresponding event profile, our goal is to create mapping $F: E \rightarrow EP$, where EP represents the event profile dataset corresponding to E and $E = \{E_n\}_{n=1}^{m}$, the entity of which is $E_i = \{e_1, e_2, e_3, \ldots e_n\}$. Hence, the number m indicates the number of rows in the dataset, and the number n is the number of event categories e_i. The dimension of the TF-IDF event set vector equals the size of n columns in the collection, which has a dependency on what kind of event occurred. Hence, whereas there could be thousands of different types, it is necessary that overfitting caused by a high dimension is reduced.

TABLE 1. Various symbols and notations used

Terms	TF-IDF For Common Text Categorization	TF-IDF for Malware Detection in Liao *et al.* [39]	Substitution of TF-IDF for our System
m	total number of documents	total number of processes	total number of event sets
n	total number of distinct words	total number of distinct system calls	total number of unique event names
n_i	number of times i th word occurs	number of times i th system call was issued	number of times i th event was issued
tf_{ij}	frequency of i th word in the j th document	frequency of i th system call in process j	frequency of i th event in event set j
D_j	j th training document	j th training process	j th training event set
X	test document	test process	test event set

For dimensionality reduction, the well-known principal component analysis (PCA) and singular value decomposition (SVD) methods are used in many deep-learning fields. However, we developed a new method based on basepoints as presented in next subsection. The primary reason is we assume that network intrusion data is broadly located in high-dimensional space. In addition, we also assume that malicious security events had high deviations among their value and they are mixed together with normal data. Particularly, we perform experiment for comparison with SVD, and the result is presented in Section VIII.

C. TRANSFORM EVENT PROFILE

In this subsection, for transforming event vectors to event profile data, we first calculate the similarity of the entire event set with each basepoint set. The basic idea of our data preprocessing to reduce the high dimensionality is to calculate the cosine similarity between each data in the collections (training data) and the data of k basepoints and the measured cosine similarities are used to characterize event patterns.

For this, in this step, our method first appoints k basepoints, the number of which is given within 0.20–0.30 percent of n, in the training data set.

To appoint k basepoints, we need to find the particular event vectors that have rarely occurred over the dataset. This is mainly because the similarity value may be diverse when comparing a rarely occurring event set with other events, while the similarity value among repeatedly occurring event sets resemble. The latter case is not effective for deep learning. Hence, for appointing k basepoints, first, the 10–20 most rare event list is prepared, the event set that contains events in the rare list, is only selected for the basepoint. Next, to reduce redundancy among basepoints, if there is a redundant basepoint after calculating similarities among k basepoints, it is substituted by another event vector. By iteratively performing this procedure, sets of k basepoints are constructed.

Next, we define a set BV, which consists of k unique basepoints with different attributes, as the reference points for measuring similarity, and calculate the cosine similarities $sim\ (E, BV)$ between each training data E and each data in the basepoint set BV. The sim(A, B) function is measured by the cosine similarity value between two event-set vectors. The cosine similarity is defined as follows:

$$sim(d,q) = \frac{d \cdot q}{\|d\|\|q\|} = \frac{\sum_{i=1}^{N} d_i q_i}{\sqrt{\sum_{i=1}^{N} d_i^2}\ \sqrt{\sum_{i=1}^{N} q_i^2}} \tag{2}$$

Given k basepoints in the form of BV = [$bv_1, bv_2, bv_3, \cdots, bv_k$], where bv_i = {$e_1, e_2, e_3, \ldots e_n$} and $bv_i \in$ E, BV $\subset$ E, and the converted dataset is the similarity matrix EP = $[ep_1,\ ep_2,\ ep_3, \cdots, ep_m\ ]$, where it is an ordered set of ep_i = { sim(E_i , bv_1), sim(E_i, bv_2), sim(E_i, bv_3) ,…, sim(E_i , bv_m) } and i is from 1 to m. The final transformed dataset EP is produced as follows:

$$E = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & & a_{3,n} \\ & \vdots & & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{bmatrix} \tag{3}$$

$$BV = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & & b_{2,n} \\ \vdots & & \ddots & \vdots \\ b_{k,1} & b_{k,2} & \cdots & b_{k,n} \end{bmatrix}$$

$$simM(E, BV^T) = simM\left(\begin{bmatrix} (E_{1,i})_{i=1}^{n} \\ (E_{2,i})_{i=1}^{n} \\ (E_{3,i})_{i=1}^{n} \\ (E_{4,i})_{i=1}^{n} \\ \vdots \\ (E_{m,i})_{i=1}^{n} \end{bmatrix}, \begin{bmatrix} (bv_{1,j})_{j=1}^{n} \\ (bv_{2,j})_{j=1}^{n} \\ (bv_{3,j})_{j=1}^{n} \\ \vdots \\ (bv_{k,j})_{j=1}^{n} \end{bmatrix}^T \right) \tag{4}$$

$$= simM\left(\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & & a_{3,n} \\ & \vdots & & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{bmatrix}, \begin{bmatrix} b_{1,1} & b_{2,1} & \cdots & b_{k,1} \\ b_{1,2} & b_{2,2} & & b_{k,2} \\ & \vdots & \ddots & \vdots \\ b_{1,n} & b_{2,n} & \cdots & b_{k,n} \end{bmatrix} \right) \tag{5}$$

$$= \begin{bmatrix} s(E_1, bv_1) & s(E_1, bv_2) & s(E_1, bv_3) & & s(E_1, bv_k) \\ s(E_2, bv_1) & s(E_2, bv_2) & s(E_2, bv_3) & \cdots & s(E_2, bv_k) \\ s(E_3, bv_1) & s(E_3, bv_2) & s(E_3, bv_3) & & s(E_3, bv_k) \\ & \vdots & & \ddots & \vdots \\ s(E_m, bv_1) & s(E_m, bv_2) & s(E_m, bv_3) & \cdots & s(E_m, bv_k) \end{bmatrix} \tag{6}$$

(s denotes cosine similarity.)*

Result matrix (6) of the similarities between each event set and k basepoints is provided to FCNN, CNN, and LSTM in the next section as an important part of the input data. In practice, the matrix data are formatted as a csv file. Each data row in one csv file becomes one input data that is fed into the first layer of artificial neural networks.

Moreover, owing to resource exhaust problem by insufficient memory of most systems, dealing with a matrix or collection with numerous data requires particular matrix operation mechanisms such as data dividing.

VI. DEEP LEARNING MODELS

In this section, we present the artificial neural networks (ANNs) that compose the AI-SIEM system. As mentioned above, our deep-learning engine consists of a multi-learning engine such as FCNN, CNN, and LSTM which are collectively named EP-ANN.

A. FCNN Model

The FCNN is the most common deep learning network, in which each node in fully connected layers is connected to every node of next layer. In an FCNN, each node is connected to all the nodes in the previous layer, and each connection has respectively different and specific weight, which is not shared by each node. In past, while the FCNN is simpler than common CNNs and RNNs, it had been known that the degrading of performance for accuracy was caused by the problem of vanishing gradient during backward propagation. However, the back-propagation problem, which had restricted the development of an artificial neural network, was resolved by the emergence of the rectified linear unit (ReLU) activation function.

Consequently, to avoid the vanishing gradient problem by Sigmoid function, most deep-learning methods generally use the ReLU activation function. We also adopted the leaky rectified linear unit (leaky ReLU) scheme as the activation function, similar to RELU. The softmax function with a cross entropy cost function at the last layer, generate the final result

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953095, IEEE Access

for each input data. The common formulas for *sigmoid*, *ReLU* and *Leaky ReLU, softmax* activation function are as follows:

$$Sigmoid = \frac{1}{1+e^{-x}} \quad (7)$$

$$ReLU = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases} \quad (8)$$

$$Leaky\ ReLU = \begin{cases} 0.01x & for\ x < 0 \\ x & for\ x \geq 0 \end{cases} \quad (9)$$

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad (10)$$

In FCNN, three-layer multi-layer perceptron (MLP) with a softmax function in the final layer is same as a multi-class logistic regression model. In general, an MLP with n hidden layers can be mathematically formulated as follows [15]:

$$H(x) = H_n(H_{n-1}(H_{n-2}(\cdots(H_{1(x)})))) \quad (11)$$

In this study, we designed and implemented the FCNN for AI-SIEM platform. Parameters for building neural networks such as the number of hidden layers, output class, and activation function types for each layer can be dynamically configured in the platform.

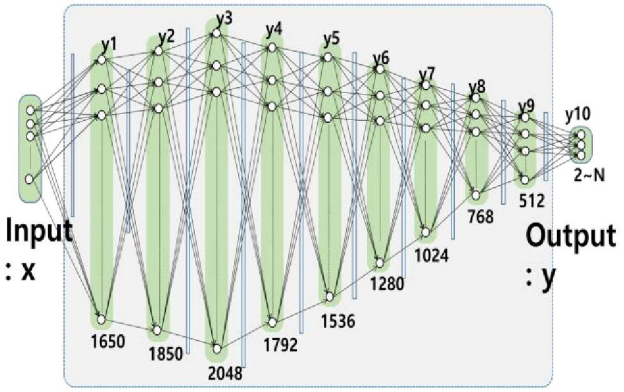

FIGURE 4. The architecture of implemented fully connected neural network (FCNN)

After performing repetitive testing, we adopted a multi-layer perceptron (MLP) model with eleven layers comprising one input layer, nine hidden layers, and an output layer. In particular, we built a suitable architecture that has one input layer and, nine hidden layers that had 1650, 1850, 2048, 1792, 1536, 1280, 1024, 768, and 512 nodes, respectively. We composed the activation functions using the leaky rectified linear unit (leaky ReLU) scheme as the activation function, instead of Sigmoid. The softmax function with a cross entropy cost function at the output layer, produces the final outputs, as shown in Figure 4. The softmax layer, which is composed of a cross-entropy cost function at the output layer, produces the final multiple outputs.

To train our FCNN, the preprocessed data were fed to the FCNN, and training was performed by tuning the parameter configuration to over 1000 epochs with a learning rate of 0.001. The implemented FCNN diagram is shown in Figure 4.

b. CNN Model

CNNs are neural network architectures especially designed to deal with spatial data. For CNN, the data of input layer consists of 2D or 3D array such as the pixel value of the image information. The core layers of CNN are convolutional layers (Conv) and max pooling layers. A Conv layer receives input as a unit and convolves it using filters to produce an ongoing data to transfer into next layers.

In a Conv layer, the filters read overall inputted data by the slicing and extract the key features. In addition, convolution is performed by calculating the scalar product between the input chunk and each filter. The features that are extracted by each filter are aggregated to a new feature set, which is called the feature map. Because the convolutional layer consists of a group of filters, it produces a feature map for each filter, and the data of feature maps are aggregated together to generate data for output [8], [22].

The designed and implemented CNN was comprised an input layer, four convolutional layers, three max pooling layers, and an output layer with one fully connected layer. Each of the front three convolutional layers in CNN was followed by max pooling layers for subsampling. We placed the dropout layer at the front of each convolutional layer except for the last.

The input layer in the implemented CNN is dynamically shaped. Because the CNN is generally specialized for 2D or 3D pixel data of the processing image, we need to transform each pre-processed event profile row into a 2D array. Hence, we transform each element of the input data vector into an N × N 2D array form, where empty positions in the 2D array are replaced with zero. Each input layer can then be variously shaped by the size of defined features for learning based on CNN. The implemented architecture for CNN is described in Figure 5, and the depicted CNN can be used to learn the data where the features ranging from number of features is 169-196.

C. LSTM Model

An LSTM has a special recurrent architecture designed to advance the storage ability.

Figure 5 presents the constructed architecture of the recurrent neural network in our deep learning model. An input layer's vector sequence $x = \{x_{t-L+1}, x_{t-L}, \ldots x_{t-1}, x_t\}$ with length L is passed with weighted connections to a layer of multiple recurrently connected hidden layers to compute first the hidden layer's vector sequences $h = \{h_{t-L+1}, h_{t-L}, \ldots h_{t-1}, h_t\}$, and then the output vector sequence $y = \{y_{t-L+1}, y_{t-L}, \ldots y_{t-1}, y_t\}$. In common LSTM, each output vector y_t is used to parameterize the probability distribution $Pr(x_{t+1}|y_t)$ of the next inputs x_{t+1} [42], [44].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953095, IEEE Access

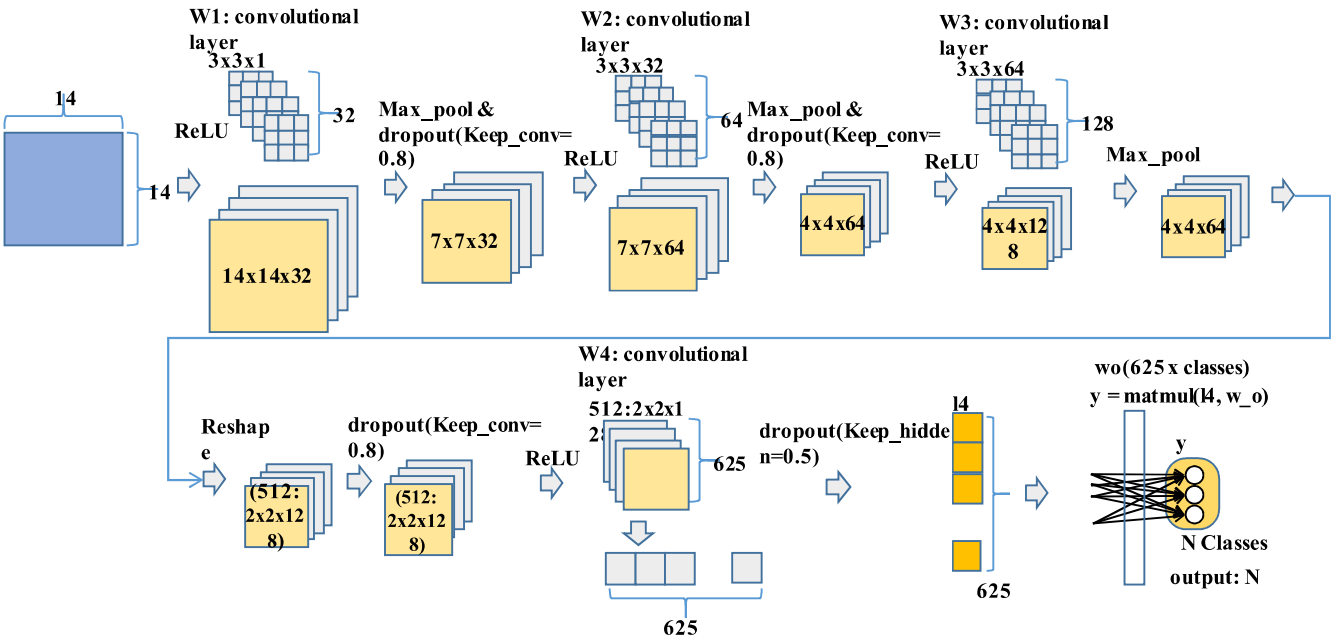

FIGURE 5. The architecture of implemented convolutional neural network (CNN).

Given the temporal dependencies between the event profiles, in this work, we employ LSTM to model the temporal correlations of event profiles. An RNN is a connectivity pattern that computes on a sequence of vectors $x_1, x_2, \cdots, x_n$, using a recurrence formula of the form $h_t = f_\theta(h_{t-1}, x_t)$, where f, an activation function and θ, a parameter, are used at each timestamp to process. To avoid the vanishing gradient problems with RNNs, gradient clipping and gating concepts are introduced [33].

An LSTM is an upgraded network of RNN. Unlike classical RNNs, LSTM tries to address the problem of long-term dependencies by introducing a purpose-built memory cell to store information of previous time steps [42].

Within this model, instead of propagating the state without multiplicative updates at each step, it is stored in memory cell C_t, which receives additive updates, merged with a method for removing irrelevant inputs from the memory cell of previous time steps [45]. Following the notation in Zaremba *et al.* [45], [46] the computation of LSTM unit at time step t is formally represented as follows:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}; x_t] + b_f) \\
g_t &= tanh(W_g \cdot [h_{t-1}; x_t] + b_g) \\
i_t &= \sigma(W_i \cdot [h_{t-1}; x_t] + b_i) \\
C_t &= f_t \odot C_{t-1} + i_t \odot g_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}; x_t] + b_o) \\
h_t &= o_t \odot tanh(C_t)
\end{aligned}
\tag{12}
$$

Here, $\odot$ represents element-wise multiplication. where x_t denotes an input vector, h_t denotes hidden state vector, C_t denotes cell state vector, o_t denotes output vector, i_t denotes input vector, and f_t denotes forget state vector, while terms W and b denote weights and biases, respectively.

Gates of memory cells consist of "input," "output," and "forget" gates. In principle, these gates enable the gradient to propagate when the model propagates through multiple steps for a long time. This is because the LSTM removes irrelevant information through the input gate i_t, memorizes information only until necessary using the forget gate f_t, and outputs only relevant information using the output gate o_t.

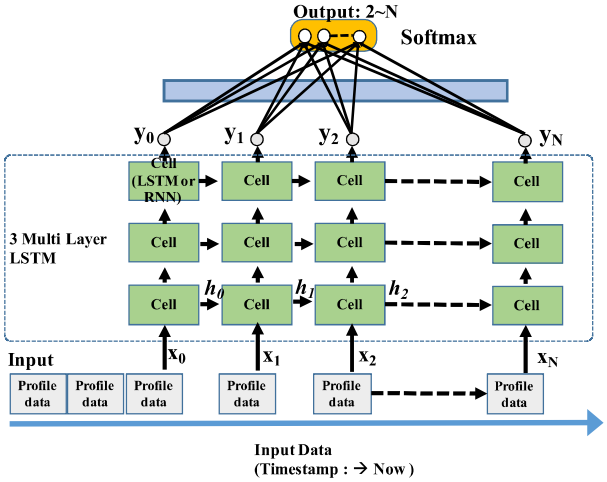

FIGURE 6. The architecture of implemented long short-term memory.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953095, IEEE Access

In our study, we constructed LSTM with 1–8 multi-layers and N hidden layers; an example of the architecture is shown in Figure 6. It must be noted that if there is one multi-layer, the neural network is an RNN. The RNN cell and LSTM cell can be easily substituted for each other because they both support in TensorFlow. To construct a suitable LSTM network with the optimal number of multi-layers and hidden layer, we used several dynamic configurations until the best performance was obtained. Consequently, we observed that the optimal number of multi-layers is 2–4 and optimal the number of hidden layers is 256–512. Although the multi-layers are deeper, this accuracy is not considerably advanced. However, a longer training period is required. Moreover, because our proposed AI-SIEM system can model the LSTM through dynamic configuration, the optimal LSTM network related to each learning data can be constructed by our system.

VII. DATASETS

This section describes the datasets. The four datasets used for testing, are NSLKDD, CICIDS 2017, and the two real datasets collected in the SOC.

A. NSLKDD

The NSLKDD dataset is the new revised version of the KDDCUP99. Tavallaee *et al.* [47] had discovered a number of duplicated records in the original KDDCUP99 dataset, which had an impact on the performance of model training and evaluation on the dataset. NSLKDD is a refined version of the dataset to address discovered statistical problems.

Some advantages over KDDCUP99 are that the complexity can be reduced and bias toward frequent records by machine learning algorithms can be prevented. However, this new version of the dataset still suffers from some of the problems discussed by McHugh in [48] and may not be a perfect representation of existing real networks. Because recent NIDS research still uses this dataset for performance evaluations, we believe it is regarded as an effective benchmark to help us compare different methods.

The training is performed on KDDTrain data which contain 22 attack types and testing is performed on KDDTest data which contains 17 additional attack types. These attacks can be categorized into four different types with some common properties for training and testing. The four categories of attacks are: Denial of Service (DoS), Probe, Remote to Local (R2L) and User to Root (U2R).

B. CICIDS 2017

In 2017, the Canadian Institute for Cybersecurity (CIC) published an intrusion detection dataset named CICIDS2017 [54]. This dataset provides the labeled data for the field of network intrusion detection research and contains benign activities and attacks, which was collected for five days log (from Monday to Friday). While the first day log contains normal activity and only includes the benign data, the other days contain the data points for various attacks together with benign data. The number of data points is approximately 2.8 million with 85 features including the label information.

The implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. The dataset has used the B-Profile system (Sharafaldin, *et al.* [49]) to profile the abstract behavior of human interactions and generate naturalistic benign background traffic.

C. REAL DATASETS

Our dataset has been collected from two large enterprise systems, named ESX-1 and ESX-2. The security raw events were collected over 5 months for ESX-1, over 30 days for ESX-2, respectively, in which the detecting threat information was separately recorded by the SOC security analysts whenever a network intrusion occurred. The list of threat detection information contains threat occurrence time, related attacks, category of attack, respond contents, attack IP address, and victim network information.

In our datasets, we investigated 798 detecting cyber threats in ESX-1, which are dispersed across the entire collection period. Looking at the type of occurred attacks in recorded cyber threats, there are 240 scanning, 547 system hacking, and 11 worm attacks. Similarly, in ESX-2 there are 941 scanning, 3,077 system hacking, and 51 worm attacks. This categorizing of attack type was manually performed by SOC analysts. By category, the system hacking attack includes a cross site script, DDoS, brute force attack, and injection attack. A trojan and backdoor attack belongs to scanning attack. Overall the number of attacks were found 4,079 cyber-threats.

TABLE 2. Distribution of Security Events in ESX-1 Dataset

ID	Prefix of Event name	count	percentage
e_2	UDP Packet Flooding	1,048,926	21.9
e_4	UDP Source-IP Flooding	718,788	15.2
e_{40}	SIP Vulnerability Scanner	644,683	13.5
e_7	TCP Connect DOS	553,362	11.6
e_{16}	TCP Invalid port	291,985	6.1
:	:	:	:
:	:	:	:
e_{15}	Psyber Streaming Server(4000/tcp)	156,750	3.3
e_7	HTTPD Overflow	115,477	2.4
e_{21}	NTP Amplification DDoS Attack BOT.B	107,617	2.3
:	:	:	:
	Total	**4,782,342**	**100**

On two datasets, we correlated each occurred attack with raw IPS security events using the above-mentioned timestamps and network information. It results that the correlated 230,026 (4.8 %) raw events are labeled as "THREAT," and the others 4,552,315 are labeled "NORMAL" in ESX-1. Moreover, in ESX-2, the correlated 1,122,636

(5.9%) raw event data are labeled as "THREAT" and 17.8 million raw event are labeled as "NORMAL."

Table 2 shows statistics of event name which collected in the ESX-1 dataset. Looking at the distribution in entire dataset, the top three events e2, e4, and e40 comprise nearly 50 percent of the collected data. The false positive rate of very frequently occurred event is relatively high, which lead to show a large amount of data to security analysts, and seriously restrict precise decision.

Table 3 presents the summarized description for NSLKDD, CICIDS 2017, ESX-1, and ESX-2 datasets which are used to evaluate the performance. Table 3 describes the summary which includes raw data collecting periods, the number of raw data, a percentage of threat (abnormal) data, and the number of attack categories.

TABLE 3. The summarized description for each dataset used to evaluate the performance

	NSLKDD	CICIDS 2017	ESX-1	ESX-2
Collecting Periods	In 1999	03/Jul/2017 – 07/Jul/2017	01/Jul/2017 – 31/Dec/2017	01/Aug/2018 – 31/Aug/2018
# of raw data (Train / Test)	148.4 K (125.9 K / 22.5K)	698 K (593 K / 105 K)	4,552 K (3,870 K / 682 K)	18,955 K (16, 112 K / 2,843 K)
Percentage of Threat Alerts in test Data	56.9 %	8.3 %	4.8 %	5.9 %
# of Attack Categories	4	7	3	3

TABLE 4. Result of event profiling of the ESX-2 dataset for different window configurations

Window interval (Sampling interval)	Window Size	ESX-2		
		# of generated event profiles (learning data)	# of Average logs in each event profile	Processing Time
60 sec	1 min	193,158	6.36	440 s
	2 min	278,157	8.95	443 s
	5 min	482,945	13.09	704 s
	10 min	782,056	16.41	1,323 s
	20 min	1,305,667	19.12	2,855 s
120 sec	1 min	96,646	6.35	381 s
	2 min	139,661	9.14	396 s
	5 min	241,796	13.10	546 s
	10 min	391,866	16.67	854 s
	20 min	646,254	20.06	1,338 s
300 sec	1 min	45,720	6.38	397 s
	2 min	61,031	8.77	378 s
	5 min	96,875	13.06	434 s
	10 min	156,413	16.65	573 s
	20 min	258,409	19.88	819 s

Table 4 shows the results of event profiling for the ESX-2 dataset for various window time intervals and sizes. The number of generated event profiles and the processing time demonstrates that when the Window Interval is increased, the number of generated event profiles and the processing time are reduced. This is because a shorter window time interval leads to further operation of the event profile processing. By contrast, if the Window Size is further increased, the number of generated event profiles and the processing time also increased. Hence, Window Interval and Window Size need to be optimally chosen for modeling. We did not determine the window interval and size for performance evaluation only based on the results of Table 4. In addition, we conducted the test by changing the window Interval and Window Size for finding the optimal values in terms of accuracy, TPR, and FPR besides the result of Table 4. For effective evaluation, we empirically applied a configuration using a Window Interval of 60 s and a Window Size of 10 min for the experiments conducted in this paper. Our proposed method aims to perform modeling by learning all the data, and consequently, we tried to perform profiling for all data without any missing portion of it. For this, all security events can be included in the event profile if and only if the window interval is less than or equal to the window size. This configuration can be modified in real environments based on the volume of data and system performances.

D. DATA VISUALIZATION WITH t-SNE

Figure 7 and Figure 8 present the distributional characteristic of the dataset used in this study. For this, we adopted t-Stochastic Nearest Neighbor (t-SNE) mechanism.

The t-SNE is not only commonly utilized for vector data visualization but also considered as embedding tools to visualize high-dimensional data. The t-SNE is able to visualize high-dimensional data into two-dimensional maps by learning two-dimensional embedding vectors that preserves neighbor structures among high-dimensional data. The N data rows in dataset are randomly selected, which are visualized by performing analysis in t-SNE [3], [50]. Figure 7 and Figure 8 represent the maps that are visualized by t-SNE for CICIDS 2017 and ESX-2, respectively. The t-SNE plots in the figure show that the normal and attack data points located nearby in the same space, which makes it very hard to classify them into either normal or attack. Although the t-SNE plots of normal and attack data are clustered, it clearly finds out that those are not linearly separated. In general, it is known that deep learning is then effective at dealing with high-dimensional data with non-linearity [51], which is one of the reasons we employ deep learning approaches to detect cyber threats.

In addition, as shown in Figure 7 and Figure 8, the data distribution visually seen by t-SNE regarding our dataset means that the dataset is not to be easily categorization in comparison with the benchmark datasets.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953095, IEEE Access

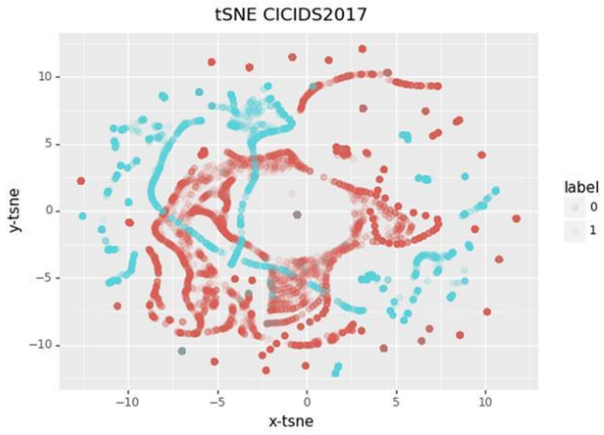

FIGURE 7. t-SNE visualization of CICIDS2017 dataset.

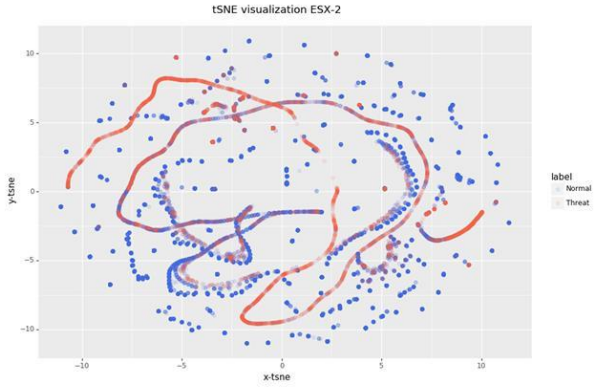

FIGURE 8. t-SNE visualization of ESX-2 dataset.

VIII. EXPERIMENTS AND RESULTS

In this section, we report the experimental results are performed with the two benchmark datasets and our two collected real datasets. We start by describing test environment with testbed. We then present the metric for experiment. Continually, we present the SVD and conventional machine-learning methods for various comparison of evaluation the performance. we discuss the experimental results in sub-section E, and finally we present the implemented system by our proposed methods.

A. TEST EVIRONMENTS

For testing, we constructed the purpose-built testbed where for conducting performance evaluations. This testbed consists of the big data platform and the AI-SIEM system. Moreover, in the SOC, we also had collected real-world IPS data over several months.

After minor data filtering, we constructed the dataset using collected data for performance evaluations as described in the previous section. In general, the format of security event of IPS/IDS is different between devices or vendors, but majority of events always contain timestamp, source ip address, destination ip address, port information, protocol, flow information, and rule names. When these security events are stored in conventional SIEM, they are stored in a standardized format with minor additions such as data tagging and data enrichment. Because the collected ESX-1, ESX-2 is a set of several types of IPS / IDS data stored through this process, it can be considered that it is sufficiently applicable to other SIEM and SOC.

For real environments when we conduct the test, we implemented a sensor emulator that can substitute for a real IPS system. It uses the syslog protocol to send to the AI-SIEM system, by reading security event dataset and synthetically generating syslog packets. For the two benchmark datasets, the sensor emulator also reads the learning data and testing data in the local system, and sends them to the AI-SIEM system.

Our proposed EP-ANN in AI-SIEM was implemented using TensorFlow [52]. The hardware used to evaluate the performance of the EP-ANN methods are clusters of server with Intel Xeon with 2.5 GHz (32 CPU cores) and 128GB memory. Two Nvidia Tesla P100 GPUs are used as the accelerator.

B. METRICS AND EXPERIMENTAL SETUP

1) FOUR METRICS

To evaluate the performance, four metrics are adopted: accuracy, TPR, FPR, and F-measure, which are all commonly used for learning-based methods in the field of intrusion detection. TPR is used to evaluate the system's performance with respect to its threat detection. FPR is used to evaluate misclassifications of normal data. F-measure is the harmonic mean of the precision and FPR(recall), where Precision= TP / (TP+FP) is the percentage of true attacks among all attacks classified, where TP (True Positive) is the number of attack data that is correctly classified as an attack, and FP (False Positive) is the number of normal data that is incorrectly classified as an attack. TN (True Negative) the number of normal data that is correctly classified as normal, and FN (False Negative) is the number of attack data that is incorrectly classified as normal. The definitions for accuracy, TPR, FPR, and F-measure are presented below:

$$TPR = \frac{TP}{TP + FN} \tag{13}$$

$$FPR\ (Recall) = \frac{FP}{TN + FP} \tag{14}$$

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{15}$$

$$F - measure = \frac{2 \cdot Precison \cdot Recall}{Precision + Recall} \tag{16}$$

2) ROC

In order to evaluate the quality of detection performance, we show a receiver operating characteristic (ROC) curve and measure an area under curve (AUC) value as significant comparison metrics.

ROC curve is a plot of FPR against TPR of binary classifiers. FPR corresponds to the proportion of normal data points incorrectly predicted as attack to all normal data points. TPR, also called sensitivity or recall, corresponds to the proportion of attack data points that are correctly predicted attack to all attack data points. ROC curve shows a trade-off between sensitivity and FPR. The closer the ROC curve is to the top-left border, the better the quality of predictions by the prediction model and vice versa [1]. Additionally, AUC is defined as area under the ROC curve, which is a measure of how well a binary classifier can perform predictions of labels. A perfect binary classifier has an AUC = 1, and a greater value of AUC shows better performance. Any AUC value less than 0.5 means poor performance of the classifier [1].

C. COMPARISON WITH SVD

As singular value decomposition (SVD) is the one of the most commonly used methods for dimensionality reduction in machine learning, we compare the performance of our method with SVD.

SVD is the method to diagonalize a matrix as in eigenvalue decomposition. Note that eigenvalue decomposition by eigenvalues and eigenvectors is applicable only to square matrices, and is also a diagonalization method applicable only to some square matrices [53]. Whereas, SVD is useful because the technique is applicable to all m × n matrices whether they are square matrices or not. SVD for an m × n matrix in real space is defined as follows:

$$A = U \times \Sigma \times V^T \tag{17}$$

where U is an *m-by-m* orthonormal matrix, V is an n-by-n orthonormal matrix and Σ is an *m-by-n* diagonal matrix. Here, an orthogonal matrix is a matrix in which the result of multiplication of itself or its transposed matrix or the result thereof is an identity matrix.

A diagonal matrix is a matrix in which the entries outside the main diagonal are all zero. The value of the diagonal element of the diagonal matrix derived from the SVD is called the singular value of matrix A. For dimensionality reduction, a *k-by-k* submatrix Σ' can be extracted from the *m-by-n* diagonal matrix Σ, and *m-by-k* submatrix U' can be extracted from *m-by-m* orthonormal matrix U.

According to SVD, the dimensionally reduced matrix *m-by-k* A' of *m-by-n* matrix A is defined as $A' = U' \times \Sigma'$, where k is the size of the reduced dimensionality for n. That is, we can obtain A' where the dimension is reduced from original dimension n to dimension k. To evaluate the performance comparison with SVD, we conducted accuracy comparison regarding the cases which the reduced dimension k of SVD is equal to the number of basepoints in our proposed method, as shown in Table 11.

D. COMPARISON WITH CONVENTIONAL ML METHODS

Before the emergence of deep learning technology, many conventional machine learning methods were adopted in intrusion detection systems for anomaly detection. Recently, it is also used in progressing. To evaluate the performance comparison with existing methods, we conducted experiments using well-known conventional machine-learning methods such as support vector machine (SVM) [54], k-nearest neighbor (k-NN) [55], random forest (RF) [56], naive Bayes (NB) [57], and decision tree (DT) [58]. Each conventional method is implemented in the WEKA library and Libsvm package [59] and all methods used the default parameters provided by the WEKA and Libsvm libraries.

E. EXPERIMENT RESULTS

In this subsection, we discuss the experimental results ~~which is~~ performed for evaluating the performance metrics such as accuracy, TPR, FPR, and F-measure. In addition, we present that the result with our proposed method achieved better performance in comparison with SVD for reducing the dimensionality space.

First, Table 5 shows the experimental result of accuracy for NSLKDD, CICIDS, ESX-1, and ESX-2 respectively. Overall, the proposed methods achieved superior performance comparison with the conventional machine-learning methods. For the NSLKDD dataset, EP-FCNN model delivered top accuracy of 0.958, EP-CNN remained runner-up in models with 0.952, in three EP-ANN model, respectively.—For CICIDS 2017, in all experimented methods except the naïve Bayes, the accuracy of each model was close to 0.98, and we could see that the performance of accuracy was similar.

Next, in Table 5, looking at our collected real ESX-1 and ESX-2 datasets, we can see that the proposed EP-ANN modes outperform the conventional existing machine-learning methods in overall experiment cases on the accuracy. In details, the result of EP-FCNN, EP-CNN and EP-LSTM achieved 0.933, 0.952, and 0.923 for ESX-1 respectively, while the experimental results of the conventional machine-learning methods remained near 0.90. For EDX-2, where the number of data is approximately four times of data in the ESX-1, the gap of performance appears a larger difference. Although both EP-FCNN and EP-CNN achieved accuracy scores of 0.947 and 0.936, the other conventional methods results near 0.85. On the whole, the overall best accuracy was delivered by the proposed EP-ANN models with accuracy score of 0.93-0.99 in four experiment datasets.

The detailed results are shown in Table 6 and Table 7. The results for benchmark dataset NSLKDD and CICIDS 2017 are presented in Table 6, and for real dataset ESX-1, ESX-2 are presented in Table 7. The objectives of testing as shown as Table 6 aims to compare our methods with conventional machine-learning methods using benchmark datasets. In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953095, IEEE Access

contrast, the testing result as shown as Table 7 aims to report whether each method is able to achieve satisfactory performance for the real data.

In Table 6, for NSLKDD, TPR was 0.905 for the k-NN, 0.891 for the RF, and 0.941 for the proposed EP-FCNN. For FPR, the EP-LSTM yielded the best performances with 0.025 (Although NB was the lowest 0.013, it is meaningless because of the lowest accuracy). Whereas 0.819 TPR of SVM is relatively low. In particular, EP-FCNN performs better than other methods for the F-Measure. For CICIDS 2017 dataset, although the performance of all the methods except Naive Bayes was almost equal, the scores TPR, FPR, and F-measure of the proposed EP-ANNs are better than others as shown in Table 6. From these results, we conclude that EP-ANNs are more effective methods for the benchmark datasets.

TABLE 5. **Test results of accuracy for various conventional machine-learning methods and our proposed.**

		Accuracy			
		NSLKDD	**CICIDS2017**	**ESX-1**	**ESX-2**
Conventional Machine Learning	**SVM**	0.897	0.968	0.901	0.867
	k-NN	0.909	0.978	0.905	0.858
	Random Forest	0.930	0.979	0.900	0.858
	Naive Bayes	0.698	0.621	0.692	0.616
	Decision Tree	0.919	0.979	0.900	0.858
Our Proposed Method	**EP-FCNN**	0.958	0.995	0.933	0.947
	EP-CNN	0.952	0.988	0.952	0.936
	EP-LSTM	0.950	0.986	0.923	0.926

TABLE 6. **Detailed Test results for various conventional machine-learning methods and our proposed methods using benchmark datasets.**

		NSLKDD				CICIDS2017			
		TPR	**FPR**	**ACC**	**F-Measure**	**TPR**	**FPR**	**ACC**	**F-Measure**
Conventional Machine Learning	**SVM**	0.819	0.035	0.897	0.881	0.925	0.023	0.968	0.912
	k-NN	0.905	0.088	0.909	0.903	0.986	0.023	0.978	0.944
	Random Forest	0.891	0.036	0.930	0.923	0.987	0.022	0.979	0.946
	Naive Bayes	0.301	0.013	0.698	0.457	0.994	0.463	0.621	0.492
	Decision Tree	0.868	0.036	0.919	0.910	0.987	0.022	0.979	0.946
Our Proposed Method	**EP-FCNN**	0.941	0.029	0.958	0.952	0.982	0.002	0.995	0.987
	EP-CNN	0.926	0.028	0.952	0.945	0.985	0.011	0.988	0.971
	EP-LSTM	0.919	0.025	0.950	0.943	0.978	0.011	0.986	0.967

TABLE 7. **Detailed Test results for various conventional machine-learning methods and our proposed methods using real datasets.**

		ESX-1 (# of raw data : 4,783,342)				ESX-2 (# of raw data 18,955,737)			
		TPR	**FPR**	**ACC**	**F-Measure**	**TPR**	**FPR**	**ACC**	**F-Measure**
Conventional Machine Learning	**SVM**	0.926	0.105	0.901	0.786	0.379	0.030	0.858	0.503
	k-NN	0.928	0.101	0.905	0.791	0.382	0.031	0.858	0.505
	Random Forest	0.926	0.106	0.900	0.785	0.382	0.031	0.858	0.505
	Naive Bayes	0.873	0.352	0.692	0.527	0.489	0.376	0.616	0.141
	Decision Tree	0.928	0.106	0.900	0.783	0.382	0.030	0.858	0.505
Our Proposed Method	**EP-FCNN**	0.885	0.059	0.933	0.781	0.899	0.049	0.947	0.688
	EP-CNN	0.902	0.041	0.952	0.833	0.895	0.061	0.936	0.643
	EP-LSTM	0.982	0.086	0.923	0.773	0.929	0.073	0.926	0.620

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2953095, IEEE Access

In Table 7, we observed that F-measure of all the methods degraded more than the results of NSLKDD and CICIDS 2017. However, for ESX-2, the TPR of the conventional machine-learning methods is close to 0.38. Whereas TPR scores of our methods retained near 0.90. Moreover, as shown in Figure 9, the result of AUC based on the ROC curve shows excellent results. The AUC values of our method were 0.967, 0.970, and 0.963 by EP-FCNN, EP-CNN, and EP-LSTM, respectively, without degradation. The AUC values for the conventional machine learning method were 0.777, 0.595, and 0.811 by SVN, NB, and DT, respectively; these exhibit remarkably degraded the performance for a large amount of data.

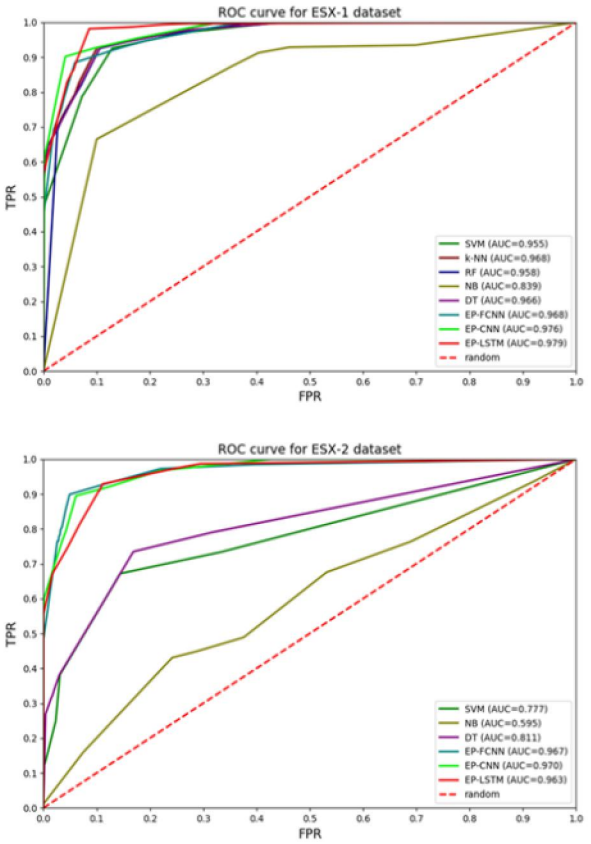

FIGURE 9. Comparison of ROC curves and AUC of experiment for ESX-1 and ESX-2 datasets.

Based on the results of this experiment, we are able to arrive at two meaningful conclusions. First, our mechanisms are capable of being employed as learning-based models for network intrusion detection. When the performance evaluations were conducted using two well-known benchmark datasets such as NSLKDD and CICIDS2017, the result proved as capable as the conventional machine-learning models. This means that our proposed methods, employed in the AI-SIEM system, have applicability for learning-based network intrusion detection. Second, when the conventional learning-based methods, which accomplish a good result by benchmark dataset, are employed in the real world, the performance of overall accuracy is not as reliable as those of benchmark datasets. Nevertheless, the accuracy performance of our three EP-ANN models were not significantly degraded, despite the large amount of data and a lack of benchmark dataset features, such as seen in the result for ESX-2. By contrast, the accuracy of conventional methods had degraded from approximately 0.90 to 0.85.

To evaluate classification performance of multi-categorization, TPR is measured for each data class of NSLKDD and ESX-2, as shown in Table 8 and Table 9, respectively. In Table 8 and Table 9, when examining detailed classification for each class, the classification accuracy for "DoS", "Normal" in NSLKDD, and "System hacking, Scanning" in ESX-2 are fairly superior. Hence, we analyze that it can't be performed sufficient data learning regarding "R2L" because there are very few, if any, data instances that are included as "R2L" type in the learning data. However, the proposed AI-SIEM system presents relatively promising results in terms of accurate classification performance, when compared with conventional machine-learning methods. In addition, we need to improve our learning methods to model, not only for major attack data, but also for infrequent attack data, as shown as Table 9.

TABLE 8. Detailed Multi-classification performance for NSLKDD

	Classification of NSLKDD (TPR)				
	DoS	**Probe**	**R2L**	**U2R**	**Normal**
SVM	0.933	0.784	0.000	0.230	0.965
k-NN	0.990	0.856	0.348	0.650	0.912
Random Forest	0.990	0.915	0.000	0.600	0.964
Naive Bayes	0.139	1.000	1.000	0.000	0.987
Decision Tree	0.979	0.851	0.001	0.550	0.964
EP-FCNN	0.984	0.834	0.210	0.050	0.971
EP-CNN	0.985	0.774	0.453	0.730	0.972
EP-LSTM	0.934	0.886	0.737	0.672	0.975

TABLE 9. Detailed Multi-classification performance for ESX-2 dataset

	Classification of ESX-2 (TPR)			
	System Hacking	**Scanning**	**Worm**	**Normal**
EP-FCNN	0.892	0.816	0.983	0.951
EP-CNN	0.897	0.918	0.920	0.939
EP-LSTM	0.931	0.912	0.000	0.927

Table 10 shows learning time and response time for the ESX-2 dataset. By this result, average response time is near 3 microseconds, which means that our system is capable of analyzing hundreds of event profiles for one second. This

capacity is considered sufficient performance to operate our system for detection in real-time.

The results obtained by our method are compared with those of the SVD, which is known as the conventional reducing dimensionality, in Table 11. Note that q indicates the number of basepoints in our method, and the reduced dimensionality in SVD. For the experiment, the ESX-1 dataset was used, and the input data for deep learning were in the same format, which consisted of m rows and q columns. From Table 11, it can be seen that our method outperforms the SVD method in each experiment.

TABLE 10. Learning time and response time for each method.

	Dataset : ESX-2	
	Learning Time	Average Response Time in Real-Time
SVM	85 m 32 s	24 ms
Random Forest	1 m 12 s	< 1 ms
EP-FCNN	28 m 12 s	4.2 ms
EP-CNN	17 m 20 s	2.3 ms
EP-LSTM	14 m 23 s	1.3 ms

TABLE 11. The result of accuracy for comparison our method with SVD

	Accuracy (dataset = ESX-1)	
	Our method	SVD
EP-FCNN (q =20)	0.912	0.726
EP-FCNN (q = 40)	0.933	0.863
EP-FCNN (q= 80)	0.943	0.856
EP-CNN (q =20)	0.912	0.751
EP-CNN (q = 40)	0.952	0.922
EP-CNN (q= 80)	0.923	0.877
EP-LSTM (q =20)	0.907	0.691
EP-LSTM (q = 40)	0.923	0.826
EP-LSTM (q= 80)	0.941	0.892

F. SYSTEM DEPLOYMENT

As explained above, the AI-SIEM system consists of event profile and artificial neural networks (EP-ANN). This system aims at protecting a number of IT systems and servers in an enterprise network, and does not have to be co-located with an IPS and the asset systems.

For system operations in practice, this is typically placed either on the access network of an enterprise with an IPS, or in the external SOC. The system is either used by a security manager of enterprise in the former case or the SOC analysts in the latter case. Thus, each module for data learning, model deployment, and real-time threat detection needs a dashboard GUI, and such a dashboard is depicted in Figure 10.

Figure 10 shows examples of dashboard screen-captures. When the AI-SIEM system detects the cyber threat using EP-ANN models, the result of the analysis is positioned in the red-zone part as shown in Figure 10-(a). Whereas, in the case of normal status, the result is depicted in the blue-zone as in Figure 10-(b). With this dashboard, the AI-SIEM system can provide intuitive monitoring for SOC analysts. Moreover, only true positive alerts detected by the AI-SIEM system are shown to the SOC security analysts through the dashboard GUI. This enables the number of alerts that need investigation to be reduced, thus decreasing the cost of false positive alerts.

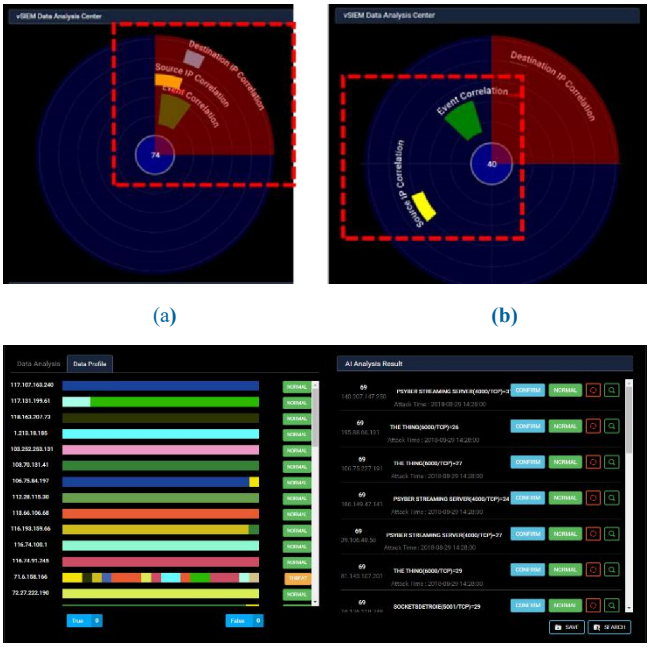

FIGURE 10. The dashboard screen-captures of the AI-based SIEM system for real-time monitoring. (a)threat detection visualization, (b) normal state visualization (c) The view for event profiles and cyber threat lists.

IX. CONCLUSION

In this paper, we have proposed the AI-SIEM system using event profiles and artificial neural networks. The novelty of our work lies in condensing very large-scale data into event profiles and using the deep learning-based detection methods for enhanced cyber-threat detection ability. The AI-SIEM system enables the security analysts to deal with significant security alerts promptly and efficiently by comparing long-term security data. By reducing false positive alerts, it can also help the security analysts to rapidly respond to cyber threats dispersed across a large number of security events.

For the evaluation of performance, we performed a performance comparison using two benchmark datasets (NSLKDD, CICIDS2017) and two datasets collected in the real world. First, based on the comparison experiment with other methods, using widely known benchmark datasets, we showed that our mechanisms can be applied as one of the

learning-based models for network intrusion detection. Second, through the evaluation using two real datasets, we presented promising results that our technology also outperformed conventional machine learning methods in terms of accurate classifications.

In the future, to address the evolving problem of cyber attacks, we will focus on enhancing earlier threat predictions through the multiple deep learning approach to discovering the long-term patterns in history data. In addition, to improve the precision of labeled dataset for supervised-learning and construct good learning datasets, many SOC analysts will make efforts directly to record labels of raw security events one by one over several months.

REFERENCES

[1] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, K. Han, "Enhanced Network Anomaly Detection Based on Deep Neural Networks," *IEEE Access*, vol. 6, pp. 48231-48246, 2018.

[2] B. Zhang, G. Hu, Z. Zhou, Y. Zhang, P. Qiao, L. Chang, "Network Intrusion Detection Based on Directed Acyclic Graph and Belief Rule Base", *ETRI Journal*, vol. 39, no. 4, pp. 592-604, Aug. 2017

[3] W. Wang, Y. Sheng and J. Wang, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, no. 99, pp. 1792-1806, 2018.

[4] M. K. Hussein, N. Bin Zainal and A. N. Jaber, "Data security analysis for DDoS defense of cloud based networks," *2015 IEEE Student Conference on Research and Development (SCOReD)*, Kuala Lumpur, 2015, pp. 305-310.

[5] S. Sandeep Sekharan, K. Kandasamy, "Profiling SIEM tools and correlation engines for security analytics," *In Proc. Int. Conf. Wireless Com., Signal Proce. and Net.(WiSPNET)*, 2017, pp. 717-721.

[6] N.Hubballiand V.Suryanarayanan,‘‘False alarm minimization techniques in signature-based intrusion detection systems: A survey,’’ *Comput. Commun.*, vol. 49, pp. 1-17, Aug. 2014.

[7] A. Naser, M. A. Majid, M. F. Zolkipli and S. Anwar, "Trusting cloud computing for personal files," *2014 International Conference on Information and Communication Technology Convergence (ICTC)*, Busan, 2014, pp. 488-489.

[8] Y. Shen, E. Mariconti, P. Vervier, and Gianluca Stringhini, "Tiresias: Predicting Security Events Through Deep Learning," *In Proc. ACM CCS 18*, Toronto, Canada, 2018, pp. 592-605.

[9] Kyle Soska and Nicolas Christin, "Automatically detecting vulnerable websites before they turn malicious,", *In Proc. USENIX Security Symposium.*, San Diego, CA, USA, 2014, pp.625-640.

[10] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, K. Li, "AI2: training a big data machine to defend," *In Proc. IEEE BigDataSecurity HPSC IDS*, New York, NY, USA, 2016, pp. 49-54

[11] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu and Ali A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," *In Proc. of the Second IEEE Int. Conf. Comp. Int. for Sec. and Def. App*., pp. 53-58, 2009.

[12] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization", *Proc. Int. Conf. Inf. Syst. Secur. Privacy*, pp. 108-116, 2018.

[13] [online] Available: http://www.takakura.com/Kyoto_data/

[14] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, pp. 41-50, Feb. 2018

[15] R. Vinayakumar, Mamoun Alazab, K. P. Soman, P. Poornachandran, Ameer Al-Nemrat and Sitalakshmi Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525-41550, Apr. 2019.

[16] W. Hu, W. Hu, S. Maybank, "Adaboost-based algorithm for network intrusion detection," *IEEE Trans. Syst. Man B Cybern.,* vol. 38, no. 2, pp. 577-583, Feb. 2008.

[17] T.-F. Yen et al., "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks", *Proc. 29th Annu. Comput. Security Appl. Conf*., New York, NY, USA, 2013, pp. 199-208.

[18] K.-O. Detken, T Rix, C Kleiner, B Hellmann, L. Renners, "Siem approach for a higher level of it security in enterprise networks", *In Proc. IDAACS*, Warsaw, Poland, 2015, pp. 322-327.

[19] en.wikipedia.org,“Security information and event management,” 2016 [Online] Available: https://en.wikipedia.org/wiki/Security-_information_and_event_management.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[21] C. Dong, C. C. Loy, K. He and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295-307, 1 Feb. 2016.

[22] A. Karpathy, "Connecting images and natural language," P*h.D. dissertation, Fac. Comput. Sci., Stanford Univ., Stanford*, CA, USA, 2016.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classificationwith deep convolutional neural networks," *In Proc. of the 25th Int. Conf.on Neural Inf. Proc. Systems -Volume 1*, ser. NIPS’12, 2012, pp. 1097-1105.

[24] Q. Zhu, X. Li, A. Conesa, and C. Pereira, “Gram-cnn: a deep learning approach with local context for named entity recognition in biomedical text,” *Bioinformatics*, vol. 34, no. 9, pp. 1547–1554, 2017.

[25] W.Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, “Malware traffic classification using convolutional neural network for representation learning,” *In Proc. Int. Conf. on Infor. Net. (ICOIN)*, Da Nang, Vietnam, Jan. 2017, pp. 712–717.

[26] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, “Intrusion detection using convolutional neural networks for representation learning,” *In Proc. Int. Conf. Neural Information Springer*, 2017, pp. 858–866.

[27] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, “Zero-day malware detection based on supervised learning algorithms of API call signatures,” *In Proc. 9th Australas. Data Mining Conf.*, vol. 121. Ballarat, Australia, Dec. 2011, pp. 171-182.

[28] E. Raff, J. Sylvester, and C. Nicholas, “Learning the PE header, malware detection with minimal domain knowledge,” *In Proc. 10th ACMWorkshop Artif. Intell. Secur*. New York, NY, USA, Nov. 2017, pp. 121-132.

[29] J. Gu et al., "Recent advances in convolutional neural networks", CoRR, pp. 187-332, Dec. 2017

[30] Kehe Wu, Zuge Chen, Wei Li, "A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks", *Access IEEE*, vol. 6, pp. 50850-50859, 2018

[31] Taejoon Kim, Sang C. Suh, Hyunjoo Kim, Jonghyun Kim and Jinoh Kim, "An Encoding Technique for CNN-based Network Anomaly Detection," *In Proc. IEEE International Conference on Big Data (IEEE BigData)*, Seattle, WA, USA, Jan. 2019, pp. 2960-2965.

[32] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran and S. Venkatraman "Robust Intelligent Malware Detection Using Deep Learning," *IEEE Access*, vol. 7, pp. 46717-46738, Apr. 2019

[33] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[34] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” 2001.

[35] D. Sisiaridis and Olivier Markowitch, "Reducing Data Complexity in Feature Extraction and Feature Selection for Big Data Security Analytics", *In Proc. Int. Conf. Data Intel. and Sec. (ICDIS)*, South Padre Island, TX, USA, May 2018, pp. 43-48.

[36] V. N. Inukollu, S. Arsi, S. R. Ravuri, "Security issues associated with big data in cloud computing", *International Journal of Network Security & Its Applications*, vol. 6, no. 3, pp. 45, 2014.

[37] Jong-Hoon Lee, Young Soo Kim, Jong Hyun Kim, Ik Kyun Kim and Ki-Jun Han "Building a big data platform for large-scale security data

analysis," *In Proc. Int. Conf. Infor. Com. Tec. Conv. (ICTC)*, Jeju, South Korea, 2017, pp. 976-980.

[38] F. A. Khan, A. Gumaei, A. Derhab and A. Hussain, "A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection," in *IEEE Access*, vol. 7, pp. 30373-30385, 2019.

[39] Min Du, Feifei Li, Guineng Zheng and Vivek Srikumar, "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning," *In Proc. ACM CCS 17*, Dallas, Texas, USA, pp. 1285-1298.

[40] Y. Liao and V. Vemuri, "Use of K-nearest neighbor classifier for intrusion detection," *Comput. Secur.,* vol. 21, no. 5, pp. 439-448, Oct. 2002.

[41] A. Oprea, Z. Li, T. Yen, S. H. Chin and S. Alrwais, "Detection of Early-Stage Enterprise Infection by Mining Large-Scale Log Data," *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Rio de Janeiro, 2015, pp. 45-56.

[42] Ke Zhang, J. Xu, M. R.Min, G. Jiang, K. Pelechrinis, and Hui Zhang. "Automated IT system failure prediction: A deep learning approach," *In Proc. IEEE Int. Conf. Big Data (IEEE BigData)*, Washington, DC, USA, Dec. 2016, pp. 1291-1300.

[43] J. Han and M. Kamber, "Data Mining Concepts and Techniques," *Morgan Kaufmann Publishers*, 02nd edition, 2006, pp. 364-365.

[44] A. Graves, "Generating sequences with recurrent neural networks," arXiv preprint arXiv:1308.0850, 2013.

[45] E. C. R. Shin, D. Song, R. Moazzezi, "Recognizing functions in binaries with neural networks," *In Proc. the 24th USENIX Security Symposium (USENIX Security '15)*, CA, USA, 2015, pp 611-626.

[46] W. Zaremba, I. Sutskever, O. Vinyals, Recurrent neural network regularization, Sep. 2014, [online] Available: http://arxiv.org/abs/1409.2329.

[47] M. Tavallaee, E. Bagheri, W. Lu, A. Ghorbani, "A detailed analysis of the KDD Cup 1999 data set", *Proc. 2nd IEEE Symp*. Comput. Intell. Secur. Defense Appl., pp. 1-6, 2009.

[48] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Trans. Inf. Syst. Security*, vol. 3, no. 4, pp. 262–294, 2000.

[49] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *In Proc. 4th Int. Conf. Inf. Syst. Secur*. Privacy (ICISSP), 2018, pp. 1-8.

[50] L. van der Maaten and G. E. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579-2605, Nov. 2008.

[51] D. Kwon, K. Natarajan, S. C. Suh, H. Kim, and J. Kim, "An empirical study on network anomaly detection using convolutional neural networks," *In Proc. 2018 IEEE 38th Int. Conf.on Dist. Comp. Systems (ICDCS*), 2018, pp. 1595–1598.

[52] M. Abadi et al. (Mar. 2016). "TensorFlow: Large-scale machine learning on heterogeneous distributed systems." [Online]. Available:https://arxiv.org/abs/1603.04467

[53] D. Kalman, "A singularly valuable decomposition: The SVD of a matrix," *College Math. J*., vol. 27, no. 1, 1996, pp. 2-23

[54] G. Wang, J. Yang, R. Li, "Imbalanced SVMBased Anomaly Detection Algorithm for Imbalanced Training Datasets", *ETRI Journal,* vol. 39, no. 5, pp. 621-631, Oct. 2017.

[55] W. Li, P. Yi, Y. Wu, L. Pan, J. Li, "A new intrusion detection system based on KNN classification algorithm in wireless sensor network", *J. Elect. Comput. Eng*., Jun. 2014.

[56] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," Procedia Comput. Sci., vol. 89, pp. 213-217, Jan. 2016.

[57] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," Inf. Secur. J., GlobalPerspective, vol. 25, nos. 1-3, pp. 18-31, 2016.

[58] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," Comput. Secur., vol. 70, pp. 255-277, Sep. 2017

[59] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," ACM transactions on Intelligent Systems and Technology (TIST), vol. 2, no. 3 p. 27, 2011.