Journal of Information Security and Applications 78 (2023) 103625

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Host-based intrusion detection with multi-datasource and deep learning

Ren-Hung Hwang [a,*], Chieh-Lun Lee [b], Ying-Dar Lin [a], Po-Chin Lin [b], Hsiao-Kuang Wu [c], Yuan-Cheng Lai [d], C.K. Chen [e]

[a] National Yang Ming Chiao Tung University, Taiwan
[b] National Chung Cheng University, Taiwan
[c] National Central University, Taiwan
[d] National Taiwan University of Science and Technology, Taiwan
[e] Cycraft Technology, Taiwan

ARTICLE INFO

Keywords:
HIDS
System logs
Network traffic
Host statistics
Multiple data sources
DL-based anomaly detection

ABSTRACT

Modern hackers display increasing sophistication. Intrusion detection systems, both network-based and host-based, now utilize machine learning for improved detection of such advanced attacks. While most of these systems rely on a single data source for training, practical scenarios often involve attack features scattered across multiple sources, posing challenges to the system's effectiveness in detection. This impairs their potential for attack detection. Thus, this study assesses three host-based data sources—network traffic, system logs, and host statistics. It evaluates and compares their combined detection capabilities across diverse attack stages and types. In the proposed framework, network traffic data is handled by a Convolutional Neural Network (CNN) for improved automatic feature selection. System log data are processed using Long Short-Term Memory (LSTM) and an attention model to enhance temporal relationship exploration. Host statistics are processed by a Deep Neural Network (DNN) to improve classification performance. Experimental results show that the F1-scores reach 1.0 for all considered attacks and attack stages when all three data sources are utilized in the detection process. Additionally, employing diverse models based on the data type leads to improved results, a fact exemplified by Lin et al. (2022) which exclusively utilized XGBoost. The host statistics were found to be highly effective in detecting attacks and were thus investigated further for different attack methods and attack stages. The results showed that the disk usage percentage (DSK), minor memory faults (MINFLT), major memory faults (MAJFLT), total virtual memory growth during the last interval (VGROW), and total resident memory growth during the last interval (RGROW) were primarily affected by all the attacks in the initial access and command and control stages. By contrast, in the impact attack stage, the affected system resources varied widely depending on the particular attack.

1. Introduction

Cyberattacks have consistently resulted in substantial losses for both individuals and businesses. Despite the existence of numerous defense mechanisms, the advancement of technology has led to increasingly complex cyberattack behaviors, continuously introducing new attack methods that can penetrate existing defense technologies. Of the many forms of attack which may be launched nowadays, advanced persistent threat (APT) attacks are particularly complex and dangerous. APTs are well-funded and traditionally associated with nation-state sponsors or very large organizations. They typically lurk within the target system for long periods of time before being activated and then conduct complex and sophisticated attacks that are hard to detect by traditional anti-virus software. APT attacks have the ability to cause great damage and thus robust systems capable of detecting their presence at the earliest stage possible are urgently required [1–3].

One of the most common methods for achieving network security is that of intrusion detection systems (IDSs). The traditional approach is the signature-based IDS, which construct a database of signature values based on known attacks and then raise an alert whenever the attack pattern matches one of the signature values stored in the database. However, while signature-based IDS provide a good performance in detecting known attacks, they cannot guard against zero-day attacks, for which no previous signature exists. Thus, anomaly-based systems have found increasing use in detecting zero-day vulnerabilities in recent years. Anomaly-based IDS record normal network behaviors and then

* Corresponding author.
E-mail addresses: rhhwang@nycu.edu.tw (R.-H. Hwang), vincent0777434@gmail.com (C.-L. Lee), ydlin@cs.nctu.edu.tw (Y.-D. Lin), pclin@cs.ccu.edu.tw (P.-C. Lin), hsiao@csie.ncu.edu.tw (H.-K. Wu), laiyc@cs.ntust.edu.tw (Y.-C. Lai), ck.chen@cycarrier.com (C.K. Chen).

https://doi.org/10.1016/j.jisa.2023.103625

Available online 12 October 2023

evaluate new data as malicious if their patterns deviate significantly from previous normal behaviors. Various machine learning (ML) methods have been proposed for learning normal data patterns, including support vector machines (SVMs) [4] and k-nearest neighbor (k-NN) [5]. Aldweesh et al. [3] showed that DL methods, such as CNN and LSTM, are more effective than traditional ML methods such as SVM and decision tree (DT) in implementing IDS systems.

DL-based IDS methods can be broadly classified as either NIDS or HIDS, depending on the data source they use. In particular, NIDS systems are deployed on the gateway of a network segment to analyze and monitor any abnormal behavior of the network packets. By contrast, HIDS systems are installed on a host system to determine whether or not it is under attack by monitoring changes in the log sources, e.g., the network traffic, system logs, host statistics, and so on. Unlike NIDS systems, which collect packet data from the entire intranet, HIDS systems collect incoming and outgoing packets from a single host. The system logs record all the events that occur on the operating system, and the host analyzes the usage of the various system resources while the system is running, such as the CPU usage, memory usage, and so forth, in order to detect possible attacks.

To effectively analyze the approaches and techniques of attacks, MITRE presented ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) [6] in 2013 to integrate and categorize the lifecycle and various stages of cyberattack behaviors and formulated 14 tactics, 188 techniques, and 379 sub-techniques in 2022. MITRE ATT&CK shows that network attacks are not a single action, but are in fact composed of multiple attack techniques applied in different stages. Therefore, the ability to recognize attacks at different stages is essential in detecting complex attacks such as APTs [7–9]. In addition, different data sources provide an effective means of detecting different attacks at different stages [10]. For example, network traffic, system logs, and host statistics provide a particularly good detection performance during the command and control(C&C), initial access, and impact stages of an attack, respectively. In general, different data sources provide different detection capabilities when under attack. For example, Mirai produces more network traffic records, while Disk Wipe generates more host statistics records. Therefore, detecting different kinds of attacks using only a single data source is extremely challenging. Compared to an IDS with a single data source, we believe a HIDS with multiple data sources can do better defense and analysis.

In recent years, many DL-based IDSs based on a single data source have been proposed, where these data sources generally take the form of network traffic [11,12], system logs [13,14], and host statistics [15,16]. However, very few studies have considered the use of multiple data sources [10,17]. Moreover, most previous methods have not considered the problem of detecting attacks at different stages (e.g., initial access, C&C, and impact). We believe that designed with multiple data sources and detection by attack stages can provide a more comprehensive understanding of attacks. Finally, even when multiple data sources have been used, they have invariably been treated simply as different features, or the same ML model has been applied to every source. In fact, however, different data sources have different characteristics, and should be handled using different DL models to improve the detection accuracy.

Accordingly, the present study proposes an integrated host-based IDS framework based on three data sources, namely network traffic, system logs, and host statistics. Depending on its data characteristics, each data source is processed by a suitable pre-processing method and then handled by a different DL model. In the testing stage, each pre-trained DL model predicts the data from different sources in the same time slot. The prediction results of the three models for each time slot are then evaluated by an ensemble method to determine the final prediction outcome for the system status. The detection ability of each data source is evaluated for seven different attacks and three different attack stages using the F1-score. The F1-score is additionally evaluated for seven different combinations of the data sources. It is shown that, in most cases, the detection results obtained using multiple data sources for prediction purposes have a higher F1-score than those which use a single data source alone.

The main contributions of this study are as follows:

- Different pre-processing techniques and DL models are used to handle different data sources in order to improve the detection performance of the respective model.
- An HIDS framework based on multiple data sources is proposed which exploits the different information and detection capabilities of diverse data sources to achieve a more comprehensive understanding and detection of sophisticated attacks.
- The F1-score is evaluated for seven different combinations of data sources in seven different attacks and three attack stages.
- The host statistics data are analyzed in order to understand the amount and type of system resources consumed by different attacks in different attack stages and hence to gain insights into the means by which the DL model is able to classify different attack types based on the host statistics.

The remainder of this paper is organized as follows. Section 2 surveys the existing state-of-the-art for anomaly intrusion detection. Section 3 defines the notations and problem considered in the present study. Section 4 introduces the pre-processing mechanisms and DL models applied to the different data sources. Section 5 presents and discusses the experimental results. Finally, Section 6 provides some brief concluding remarks and indicates the intended direction of future research.

2. Related work

Table 1 summarizes the current state-of-the-art ML-based IDS methods, where these methods are classified into four categories depending on their data source, namely network traffic, system logs, host statistics, and multiple sources.

2.1. Network traffic

Many studies have proposed DL models for abnormal network traffic detection. Generally speaking, these models use unsupervised learning methods to detect new attacks by extracting the features automatically from the network traffic. For example, Li et al. [11] used random forest (RF) to filter out the critical features of the network traffic and then used these features together with an autoencoder to perform training. In a previous study [12], the present group used a CNN to extract the features of the network traffic and then used an autoencoder to train the extracted features. Notably, for each flow, only the first few packets (packed into segments with a fixed length of 60 bytes) were processed, and hence the computation time and memory space were greatly reduced compared to that of statistical-based feature extraction methods [11,18]. In addition to autoencoder, CNN is also suitable for training complex data. Zhang et al. [18] used the SMOTE-ENN algorithm to solve the imbalanced data issue inherent in network traffic datasets and converted the resulting data into an image format for CNN training purposes. Zeng et al. [19] proposed an integrated framework designated as Deep-Full-Range for intrusion detection based on three different DL models, namely CNN for learning the features of the raw traffic in the spatial range, LSTM for learning the features in the temporal domain, and stacked autoencoder (SAE) for extracting the features from the coding characteristics. Sun et al. [20] proposed a hybrid network for intrusion detection, in which CNN was used for feature extraction and LSTM for training. The present group recently proposed a novel packet-level IDS based on LSTM [21].

Table 1
ML-based anomaly intrusion detection approaches.

Paper	Data Sources			Dataset	Algorithm	Performance
	Network traffic	System logs	Host statistics			
[11]	V			CSE-CIC-IDS 2018	RF+Autoencoder	0.53~1.00 (AUC)
[12]	V			USTC-TFC 2016	CNN+Autoencoder	0.99~1.00 (F1)
[18]	V			NSL-KDD	SMOTE-ENN+CNN	83% (Acc)
[19]	V			ISCX VPN-nonVPN, ISCX 2012	CNN+LSTM+SAE	0.96~1.00 (F1)
[20]	V			CICIDS2017	CNN+LSTM	98.67% (Acc)
[21]	V			ISCX2012, USTC-TFC2016, Mirai-RGU, Self-collected	LSTM	0.97~1.00 (F1)
[13]		V		HDFS, OpenStack	LSTM	0.96~0.98 (F1)
[14]		V		HDFS, BGL	Attention-based LSTM	0.95~0.96 (F1)
[22]		V		HDFS, self-collected	Attention-based Bi-LSTM	0.91~0.99 (F1)
[23]		V		Self-collected	CNN+LSTM	78.6% (Acc)
[24]		V		HDFS, BGL	multi-head self-attention	0.94~0.96 (F1)
[25]		V		HDFS, BGL	embedding+LSTM+ self attention	93%~99% (Acc)
[15]			V	Self-collected	One-class K-means+ Univariate Gaussian	91% (Acc)
[16]			V	Self-collected	SVM	99% (Acc)
[26]			V	Self-collected	SVM+TFPG	98.71% (Acc)
[27]		V	V	Self-collected	RF	100% (Acc)
[17]	V		V	Self-collected	OneR, DT, NB, BN, LR, RF, KNN, SVM, K-means	91%~100% (Acc)
[10]	V	V	V	Self-collected	XGBoost	0.97 (F1)
Ours	V	V	V	CR»ME	CNN, DNN, LSTM+self-attention	1.00 (F1)

2.2. System logs

Log files record all the events in the operating system in a text format. Sequence DL models such as RNN and LSTM are well suited to natural language processing (NLP) problems, in which the aim is to learn the temporal relationships among sentences. Thus, many IDSs which use system logs as the data source employ sequence models as the DL model. For example, in the DeepLog system proposed in DeepLog [13], LSTM is used to automatically learn the log patterns under normal execution and to detect anomalies when the log patterns deviate from the learned model. In the LogAnomaly framework proposed in [14], the original log files are first converted into templates using the FT-Tree log parser, and the templates are then converted into vectors using DLCE. Finally, the log file data are trained using an Attention-based LSTM. Once a certain amount of new data have been collected, they are added to the dataset, and the model is retrained. In the LogRobust system [22], the original logs are converted into vectors using the Drain log parser and Fast-Text tools, and TF-IDF is then employed to determine the relative importance of the data items and adjust the weights accordingly. Finally, the dataset is trained using an Attention-based Bi-LSTM. Tan et al. [23] converted the system log data into 2D matrices using a word embedding approach and then employed a CNN-LSTM hybrid DL model to extract the features and train the model. Wang et al. [24] used BERT to extract the semantic vectors of the data log, and then employed a multi-head self-attention mechanism to learn the context information of the log sequence. Yang et al. [25] proposed an intrusion framework based on LSTM with self-attention and showed that the framework not only improved the detection accuracy compared to previous methods, but also reduced the training time and testing time.

2.3. Host statistics

Most IDSs that use host statistics as the training data are designed for IoT equipment and mobile devices (e.g., Android). Since such devices usually have only limited computing resources, the IDS systems are generally implemented using simple ML or statistical methods. For example, Ribeiro et al. [15] used a univariate Gaussian algorithm and K-means algorithm to analyze and evaluate the host statistics of Android. Ham et al. [16] used SVM as a classifier due to its ability to filter out unnecessary features automatically by reducing the dimensionality. Sun et al. [26] employed a two-stage detection scheme using SVM and the Temporal Failure Propagation Graph (TFPG) algorithm, designed for smart meter applications, effectively reducing the false alarm ratio and conserving computational resources.

2.4. Multiple data sources

Several previous IDS systems use multiple data sources. For example, the E-Spion system in [27] provides a three-layer intrusion host-based detection service at an edge server for the Internet of Things (IoT) devices by monitoring the white-list of all running processes in one layer and detecting the system call and host statistics using an RF approach in the other layers. Ribeiro et al. [17] presented an IDS system for Android mobile devices based on network traffic and host statistics and evaluated the performance of many different ML methods, including One Rule (OneR), DecisionTree (DT), Naïve Bayes (NB), BayesianNetwork (BN), Logistic Regression (LR), RF, KNN, SVM, and K-means. Lin et al. [10] used XGBoost to train three different data sources, namely system logs, network traffic, and host statistics, and then compared the detection performance of seven different combinations of these data sources. In general, the results showed that using multiple data sources to detect malware was more effective than using single data sources alone.

Although some previous studies used multiple data sources, they either treated different data sources as a single type of data source or used the same ML model to handle all of the data sources. However, intuitively, the detection performance of IDS systems is likely to be improved by utilizing different DL models to learn the features of the different data sources (e.g., sequence models to learn the temporal relationships between the sentences in system logs, CNN to learn the features of network traffic, and so on). This motivates the present work to adopt different DL models for different data sources.

Table 2
Notation table.

Category	Notation	Description
Datasets	D^{nt}	Network traffic dataset
	D^{sl}	System logs dataset
	D^{hs}	Host statistics dataset
Models	M^{nt}	Pre-trained network traffic model
	M^{sl}	Pre-trained system logs model
	M^{hs}	Pre-trained host statistics model

3. System model and problem formulation

This section defines the notations used in the present study and introduces the considered problem.

3.1. Notations

Table 2 defines the notations used in the present work. In the proposed framework, three data sources are collected from [28]: the network traffic dataset D^{nf}, the system logs dataset D^{sl}, and the host statistics dataset D^{hs}. The three data sources are pre-processed in accordance with their data type and, according to the timestamp, 80% of the data in each data source are taken as training data, while the remaining 20% are retained as testing data. The pre-processed data are used to train three DL models, where these models are chosen in accordance with the particular characteristics of the corresponding data source. The training process yields three trained models, namely the pre-trained network traffic model M^{nt}, the pre-trained system logs model M^{sl}, and the pre-trained host statistics model M^{hs}. Finally, the testing data are applied to each of the pre-trained models for prediction purposes, and the ensemble method is used to combine the prediction results of each model and generate the overall predicted state of the network (normal or under attack).

3.2. Problem statement

- **Input:** Labeled D^{nt}, labeled D^{sl}, and labeled D^{hs}.
- **Output:** Label prediction (benign or malicious).
- **Objective:** Optimize performance of malicious behavior detection on a host using deep learning method.
  - Performance metric: F1-score
- **Constraint:** Test based on a combination of different data sources.

As described in Section 3.1, the inputs to the proposed IDS framework comprised three host-based datasets, namely D^{nt}, D^{sl}, and D^{hs}. A training process was performed using different DL models to obtain pre-trained models M^{nt}, M^{sl}, and M^{hs}, respectively. Finally, an ensemble approach was used to predict the network status in each time slot as either benign or malicious. Experiments were conducted to compare the F1-scores of seven different combinations of the data sources across seven attacks and three attack stages. (Note that the seven combinations of data sources consisted of the three individual data sources, three combinations of two data sources, and one combination of all three data sources.)

4. Multi-datasource multi-stage deep learning framework for malicious intrusion detection

4.1. Overview

The present study develops a DL-based HIDS system and evaluates its detection capabilities for seven different combinations of data sources. Fig. 1 presents a schematic overview of the proposed framework. As shown, three data sources, D^{nt}, D^{sl}, and D^{hs}, are collected from CR»ME [28], a toolchain capable of simulating various attacks. The datasets are pre-processed using different pre-processing techniques chosen in accordance with the data characteristics of the corresponding data source. For each data source, 80% of the pre-processed data are selected for DL model training. To improve the detection performance, the pre-processed data from the different data sources are trained using different state-of-the-art DL models in accordance with their particular characteristics. The testing data are sorted by time window and applied to the trained DL models for prediction purposes. Finally, the ensemble method is employed to predict the state of each time slot as either benign or malicious. The detection performance of the proposed method is evaluated for seven attack modes and three attack stages using the F1-score metric.

4.2. Data preprocessing

Three host-based datasets (D^{nt}, D^{sl}, and D^{hs}) were collected from the CREME testbed [28]. Before feeding the data sources to the DL models, they were pre-processed using the methods described in the following.

Network traffic: High-speed network environments usually generate massive amounts of network traffic. Thus, in a previous study by the present group [12], a pre-processing stage was applied to reduce the burden of collecting and detecting the packets and speed up the processing time by extracting only a few packets from each flow. The present study adopts a similar approach. In particular, the PCAP files collected from CREME were converted into flows in accordance with their 5-tuple information (source IP, destination IP, source port, destination port, and protocol). If the duration of a flow was longer than that of a time slot (default time: 1 s), the flow was cut into sub-flows. The CREME flows were labeled as either benign or malicious based on their source and destination IPs; thus, the IP addresses were anonymized. Finally, as in [12], a fixed amount of data (e.g., the first 60 bytes of the first three packets) was extracted from each flow. If the number of packets in the flow was less than three, or the total number of bytes belonging to a packet was less than 60, the data was padded with zeros. The extracted 180 bytes were then converted into a one-dimensional gray-level image consisting of 180 pixels.

System logs: CR»ME converts the original log files into templates using the parsing tool Drain [29]. In the present study, the words in these templates were processed by Word2vec to produce an n-dimensional vector (default n = 100) from which the semantic relationships among the words were then learned. In particular, each template was transformed into a two-dimensional array, where the first dimension represented the word index and the second dimension was the corresponding n-dimensional word vector. The TF-IDF method described in [30] was then used to determine the importance of each word in the vector of templates in order to adjust their weights and merge them into a single n-dimensional vector. The sequence length was taken as a fixed window with a size of 11. In other words, eleven templates were combined into one vector. If the number of templates was insufficient, the sequence was simply padded with zeros.

Host statistics: The host statistics indicate the status of the various system resources during each process, such as the CPU usage and memory usage (see Table 3). In the present study, each process was considered as a data point, and multiple system resources were associated with each point. Each system resource usage statistic was thus initially converted to a form that the DL model could identify, e.g., 15% to 0.15, 42 K to 42,000, and so on. After the conversion process, each data point was represented as a vector, in which each feature represented the usage status of a particular system resource. Finally, min–max normalization was performed to convert each data value to the range of 0 to 1.

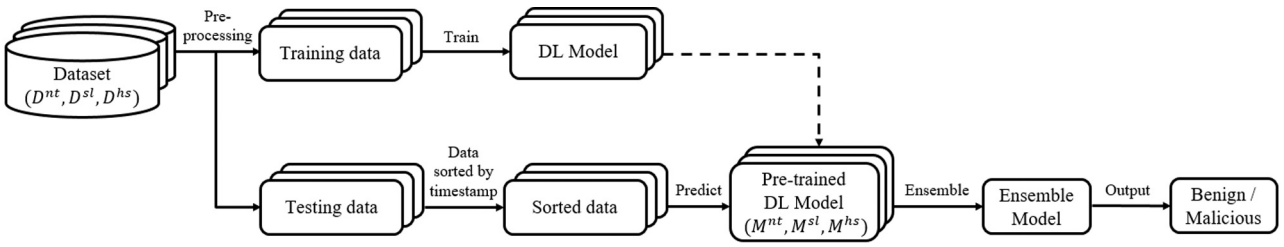


Fig. 1. Overview of system structure.

Table 3
Definition of host statistics features.

Feature	Feature name	Description
1	RDDSK	The number of read data transfer issued physically on disk.
2	WRDSK	The number of write data transfer issued physically on disk.
3	WCANCL	The number of write data been removed before transfer issued physically on disk.
4	DSK	The percentage of total disk accesses.
5	MINFLT	The number of page faults has been solved by reclaiming the requested memory page from the free list of pages.
6	MAJFLT	The number of page faults has been solved by creating/loading the requested memory page.
7	VSTEXT	The virtual memory size used by the shared text.
8	VSIZE	Total virtual memory usage.
9	RSIZE	Total resident memory usage
10	VGROW	The amount of virtual memory that the process has grown during the last interval.
11	RGROW	The amount of resident memory that the process has grown during the last interval.
12	MEM	Memory usage.
13	TRUN	Number of threads within this process that are in the state 'running' (R).
14	CPU	CPU utilization.

4.3. Deep learning models

The pre-processed data sources (network traffic, system logs, and host statistics) were handled using three different DL models, as described in the following.

Network traffic: CNNs provide an outstanding performance for many computer vision and image processing tasks, such as image classification [31]. Moreover, many studies have applied 1D-CNNs to analyze network traffic flows [12,19]. The superior performance of CNNs in such tasks stems from their ability to automatically extract and learn the useful features from the data. Accordingly, in the present study, a CNN was deliberately chosen as the DL model for network traffic training. In particular, the one-dimensional images of the network traffic data (each with a size of 180 pixels) were taken as the input of the CNN and were processed by two convolutional layers followed by a max pooling layer. A batch normalization operation was performed after each convolutional layer to avoid overfitting and the vanishing gradient problem. Following the max pooling layer, a flattened layer and three dense layers were used to conduct learning. ReLU was used as the activation function in all of the layers other than the last dense layer, which used the sigmoid function. Finally, binary cross-entropy is used as the loss function. The aim of the CNN model was to classify each flow as either benign or malicious; thus, the final output of the model had the form of either zero or one, where zero indicated that the flow was benign and one indicated that the flow was malicious. The architecture of the CNN model is shown in Table 4.

Table 4
Network traffic DL model architecture.

Layer	Type	Filters/neurons	Stride	Padding
1	1D/ConV+Relu+ Batch Normalization	32 (kernel size= 6)	1	5
2	Maxpooling	Kernel size = 2	2	–
3	1D/ConV+Relu+ Batch Normalization	64 (kernel size= 6)	1	5
4	Maxpooling	Kernel size = 2	2	–
5	Dense+ Batch Normalization	1024	–	–
6	Dense+ Batch Normalization	25	–	–
7	Dense	1	–	–

Table 5
System logs DL model architecture.

Layer	Type	Output shape	Parameter#
1	LSTM	(None, 11, 64)	67840
2	SeqSelfAttention	(None, 11, 64)	4097
3	GlobalAveragePooling1D	(None, 64)	0
4	Dense	(None, 1)	65

System logs: The system logs dataset was handled using a sequence model due to its ability to learn the temporal relationships among sentences. Vaswani et al. [32] proposed the transformer approach. The most important mechanism of the transformer is self-attention, which is able to check attention with all words in same sentence at once, and the performance is better than the traditional attention approach. Table 5 shows the sequence model implemented in the present study. As shown, the first and second layers consisted of a LSTM and self-attention model, respectively. The output of the self-attention model was processed by a global average pooling 1D layer to return a fixed-length output vector for each example by averaging the sequence dimension. Finally, a dense layer was employed to learn binary classification. In implementing the learning process, the sigmoid function was used as the activation function, and binary cross-entropy was used as the loss function.

Host statistics: Most previous IDS studies based on host statistics focus on IoT or mobile devices and use traditional ML or statistical methods due to the limited computing resources of such devices [15,16]. However, preliminary experiments conducted in the present study showed that a DNN model outperformed these methods in handling host statistics data. Thus, the present study adopted the DNN model shown in Table 6, consisting of five dense layers. ReLU was used as the activation function in the first four layers, while sigmoid was used in the final layer. The model was implemented using the binary cross-entropy loss function.

Table 6
Host statistics DL model architecture.

Layer	Type	Output shape	Parameter#
1	Dense	(None, 32)	512
2	Dense	(None, 64)	2112
3	Dense	(None, 128)	8320
4	Dense	(None, 256)	33024
5	Dense	(None, 1)	257

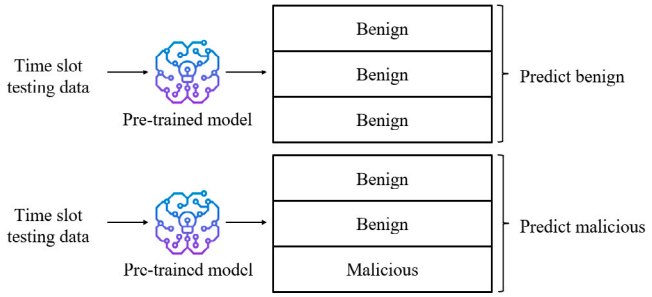


Fig. 2. Single model prediction.

4.4. Ensemble

To evaluate the detection performance of the proposed framework for different combinations of data sources, the ensemble method was used to combine the prediction results of each model for the network status (i.e., normal or under attack) in each time slot. As described in Section 3.1., for each model, the data in the corresponding dataset (D^{nt}, D^{sl}, or D^{hs}) was divided into a training set and a test set according to the timestamp. For example, timestamps 1637482171 to 1637482174 were chosen as training data, while timestamp 1637482175 was chosen as testing data. The training sets were used to train the DL models of the three data sources (M^{nt}, M^{sl}, and M^{hs}). The testing sets were then used to generate predictions of the network status in each time slot. Finally, the predictions of the three models were ensembled to make a final prediction of the network status.

Fig. 2 illustrates the time slot prediction process of each DL model. In practice, each time slot may contain several data points, and each data point may yield a different prediction result. Thus, the final prediction result of the model is based on the presence (or otherwise) of a malicious prediction among all the data points. That is, if even one of the data points in the time slot is judged to be malicious, the entire time slot is deemed to be malicious. Conversely, if all of the data points in the time slot are predicted to be normal, the time slot is classified as benign.

Since the three data sources are partitioned into the training set and testing set using the same timestamps, the predictions of the three models for each timestamp are aligned with one another, as shown in Fig. 3. If one of the data sources does not have any data points in a time slot, system status of that data source is assumed to be benign in that time slot. Under the ensemble scheme considered in the present study, if all the pre-trained models predict the time slot to be benign, the time slot is assumed to be benign. However, if even one of the pre-trained models predicts the time slot to be malicious, then the time slot is considered to be malicious. The rationale for this approach lies in the fact that an attack generally results in only very few data points. Thus, if a malicious data point appears in a time slot, it is likely that an attack occurred in that time slot, irrespective of the prediction outcomes of the other models.

	Traffic	Log	Statistics	Ground Truth
Time slot 1	Benign	Benign	Benign	Benign
Time slot 2	Benign	Benign	Malicious	Malicious
Time slot 3	Malicious	Malicious	Benign	Malicious
Time slot 4	Benign	X	Benign	Benign
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Time slot N	X	Benign	Malicious	Malicious

Fig. 3. Prediction outcomes of multiple models in each time slot.

5. Evaluation

5.1. Experimental design

In this section, we describe the experimental environment, the content of the dataset used in the experiments, and the evaluation metrics.

5.1.1. Experimental setup

The experiments were conducted on a server-class machine with an Intel(R) Xeon(R) Silver 4116 CPU, Nvidia GV100GL GPU, and 256G memory. The DL models were implemented using TensorFlow 2.3.0 and Keras 2.4.0, with CUDA version 11.0. The operating system was Ubuntu 18.04.

5.1.2. Datasets

CR»ME is a toolchain that emulates seven types of attacks and automatically collects the simulated data [28]. The collected data include network traffic, system logs, and host statistics, where these data are labeled as either normal or abnormal. CR»ME is hosted on a set of virtual machines (VMs) in an emulated attack environment and can run on Windows or Linux. The benign and target servers run on Metasploitable 3 (Ubuntu 14.04), emulating multiple legitimate services such as Web, Email, and FTP, while the attack hosts run on Kali Linux. The host-based data used in the present study were collected from a single VM which emulated both benign and malicious behaviors.

CR»ME can emulate seven attacks: Mirai, Ransomware, Resource Hijacking (Mining), Disk Wipe, Endpoint Denial of Service (endpoint DoS), Data Theft, and Rootkit Ransomware. Each attack consists of three stages: initial access, C&C, and impact, according to ATT&CK, and each attack exhibits different behaviors at different stages. The different data sources (D^{nt}, D^{sl}, D^{hs}) were collected, and their features, extracted using different tools. For example, the network traffic was gathered as PCAP files using *tcpdump*, and the PCAP files were converted into flow-based data using *Argus*. Meanwhile the system logs were collected using *rsyslog* and were converted into templates using *drain*. Finally, the host statistics were collected and classified using *atop*.

5.1.3. Evaluation metrics

We use F1-score as the metric to evaluate the performance of proposed DL models. F1-score can be calculated from the confusion matrix which is shown in Table 7.

Precision: $\frac{TP}{TP+FP}$. Percentage of all data predicted to be malicious by the model that is actually malicious.

Recall: $\frac{TP}{TP+FN}$. Percentage of all malicious data correctly predicted to be malicious.

F1-score: $\frac{2*Precision*Recall}{Precision+Recall}$. Harmonic mean of precision and recall. A higher F1-score indicates a more stable model.

Table 7
Confusion matrix for intrusion classification.

		Predict	
		Malicious	Benign
Actual	Malicious	True Positive (TP)	False Negative (FN)
	Benign	False Positive (FP)	True Negative (TN)

5.2. Detection performance of single data source and multiple data source models for different attacks and attack stages

This section compares the detection capabilities of the proposed framework (see Fig. 1) with seven combinations of the data sources for different attack stages and attack methods. Table 8 shows the number of malicious data points collected from CR»ME for each of the seven attack modes. Note that, for each attack mode, the number of logs, traffic, and statistics data points indicates the number of malicious templates, flows, and processes, respectively. It can be seen that the data points are non-uniformly distributed over the three data sources, with the number of host statistics data points being far higher than that of the number of system logs or network traffic data points.

5.2.1. F1-scores for different attack stages

Table 9 shows the F1-scores obtained for the seven attacks using the different combinations of data sources. It is seen that the F1-score is equal to 1 when all three data sources are employed, or when the statistic or logs data source are employed alone. The F1-score also has a value of 1 when the network traffic and statistics data sources are combined, or the log and statistics data sources are combined. For the network traffic data source alone, the overall F1-score is 0.93. However, the score increases slightly to 0.951 when the network traffic is combined with the log data. In terms of the detection capability in the three attack stages, the F1-score has a value of 1 in all three stages for all of the considered data source combinations other than those involving the log and network traffic data or traffic data alone. Among the latter two combinations, the combined use of the log and network traffic data yields a slightly higher F1-score than that obtained using the traffic data alone. The network traffic data source achieves a reasonable performance in the C&C stage (0.96), a poor performance in the initial access stage (0.917), and an intermediate performance in the impact stage (0.919). However, the host statistics and log data sources each achieve an F1-score of 1 in all three attack stages. Interestingly, the results indicate that the use of two data sources does not necessarily guarantee a better detection performance than that obtained using a single data source. For example, the system logs data source yields an overall F1-score of 1, but the F1-score reduces to 0.951 when it is combined with the network traffic data source. It is speculated that this performance degradation is due to missing data points. In particular, some data sources with a superior detection capability have no data points in certain time slots, and hence cannot help predict the status of the time slot (benign or malicious). In such a case, the prediction process relies on the outcomes of those data source which do have data points in the time slot, even though they may have a poorer detection capability than the original (i.e., superior) data source.

5.2.2. F1-scores of different attacks

Table 10 shows the F1-scores obtained for the seven attacks using the different combinations of data sources. As for the staged-based results presented in Table 9, it is seen that all three data sources are able to predict some (or all) of the attacks. Furthermore, even in the case where one or two data sources are unable to predict the attack with absolute reliability, the F1-score can still reach 1 when all of the data sources are combined.

5.2.3. F1-scores for different attacks at different stages

Table 11 shows F1-scores obtained by the various data sources in the seven attacks and three attack stages. Note that a blank field indicates that no malicious data were collected in the corresponding stage, and hence the F1-score could not be calculated. Moreover, the All row at the foot of the table shows the F1-score obtained using all of the testing data in the corresponding attack stage. It is noted that some of the data sources have an F1-score of 1 for one or more of the attacks, but a score of less than 1 in the All row since some of the blank fields of the benign data are incorrectly predicted. It is also observed that the system logs did not contain any malicious data points in some attack stages of several attacks. By contrast, the network traffic data source and host statistics data source collected more attack behaviors. In other words, the results confirm the need to combine multiple data sources to compensate for the fact that some data sources may fail to reveal the presence of malicious behavior in some time slots. To further examine this phenomenon, Table 12 shows the percentage of abnormal time slots correctly predicted by the different ensemble models (referred to hereafter as the attack detection rate). From Table 12, we can observe that system logs have a relatively lower attack detection rate as it observes fewer malicious data points.

5.3. Performance comparison: XGBoost vs. DL

The performance of the proposed detection framework was further evaluated by comparing the F1-scores obtained using different combinations of the three data sources with those obtained when applying the method in [10] to the same dataset. The method in [10] also utilizes different combinations of data sources to detect malicious behavior in different attack stages and attack methods. However, in contrast to the method proposed in the present study, all of the data sources in [10] are handled using XGBoost as the ML model. Figs. 4 and 5 compare the F1-scores of the two methods in different attack stages and attack modes, respectively. It is seen that the proposed method significantly outperforms that in [10] in most cases and has a comparable performance in the remaining cases. In other words, the results confirm the importance of choosing an appropriate DL model to handle each data source in accordance with its particular characteristics.

5.4. Model efficiency and feature analysis

Since the size of a data source is different at each time slot, we measure the average inference time of an ML model by dividing the amount of time it takes to make predictions for the testing dataset by the total number of seconds of the dataset. The averaged inference times of the network traffic, system logs, and host statistics data sources were found to be 0.006, 0.004, and 0.021 s, respectively. Even for the case where three data sources are used in the detection process, the average inference time is still not very long (i.e., less than 0.023 s per time slot (1 s)). In other words, the proposed framework offers the potential for the real-time detection of host attacks.

5.4.1. Feature analysis of host statistics

The results presented in the preceding sections have shown that, among the three data sources, the host statistics data source provides a larger number of malicious data points and provides the most robust detection ability. Consequently, a further detailed investigation was performed to determine the usage status of the various system resources (i.e., the host statistics features) under various malicious attacks and normal behaviors. Table 3 defines each of the 14 host statistics features. Detailed discussions can be found in Appendix.

In summary, the results presented in Appendix show that different attack behaviors affect the usage of different system resources. In the initial access and C&C attack stages, the primary system resources affected are MINFLT, MAJFLT, VGROW, and RGROW for all of the attack methods. However, in the impact attack stage, different system resources are affected, depending on the particular type of attack. Furthermore, a significant difference in the system resource usage patterns exists between the attack processes and the benign processes.

Table 8
Malicious data points.

	Initial access			C&C			Impact		
	Logs	Traffic	Statistics	Logs	Traffic	Statistics	Logs	Traffic	Statistics
Mirai	0	1	18	0	0	0	24	287	579
Mining	1	9	249	1	14	185	83	10	1145
Ransomware	12	86	1645	2	14	942	9	8	4213
Disk wipe	80	10	288	5	10	485	160	4	1461
End point DoS	91	14	506	0	0	0	327	11	2713
Rootkit ransomware	0	128	347	3	65	752	7	56	6663
Data theft	0	17	208	83	18	207	81	14	1384
All	184	265	3261	94	121	2571	691	390	18158

Table 9
F1-score of stage-based comparison.

Overall	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Statistic	1.000
Log	1.000
Log + Traffic	0.951
Traffic	0.930

Initial Access	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Statistic	1.000
Log	1.000
Log + Traffic	0.929
Traffic	0.917

Command and Control	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Statistic	1.000
Log	1.000
Log + Traffic	0.960
Traffic	0.960

Impact	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Statistic	1.000
Log	1.000
Log + Traffic	0.960
Traffic	0.919

Table 10
F1-score comparison based on attack type.

Mirai	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Statistic	1.000
Log	1.000
Log + Traffic	0.824
Traffic	0.824

Mining	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Statistic	1.000
Log	1.000
Log + Traffic	0.889
Traffic	0.889

Ransomware	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Log + Traffic	1.000
Statistic	1.000
Traffic	1.000
Log	1.000

Disk Wipe	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Log + Traffic	1.000
Statistic	1.000
Traffic	1.000
Log	1.000

Endpoint DoS	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Log + Traffic	1.000
Statistic	1.000
Log	1.000
Traffic	0.889

Rootkit Ransomware	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Statistic	1.000
Log	1.000
Log + Traffic	0.960
Traffic	0.960

Data Theft	
Combination	F1-score
All	1.000
Traffic + Statistic	1.000
Log + Statistic	1.000
Log + Traffic	1.000
Statistic	1.000
Traffic	1.000
Log	1.000

Table 11
F1-scores of testing data for different attacks and attack stages.

	Initial access			C&C			Impact		
	Logs	Traffic	Statistics	Logs	Traffic	Statistics	Logs	Traffic	Statistics
Mirai			1.000				1.000	0.933	1.000
Mining			1.000		0.857	1.000		1.000	1.000
Ransomware		1.000	1.000		1.000	1.000	1.000		1.000
Disk wipe	1.000	1.000	1.000		1.000	1.000			1.000
Endpoint DoS	1.000	1.000	1.000				1.000	0.857	1.000
Rootkit ransomware		1.000	1.000	1.000	1.000	1.000		0.909	1.000
Data theft		1.000	1.000		1.000	1.000	1.000	1.000	1.000
All	1.000	0.917	1.000	1.000	0.960	1.000	1.000	0.919	1.000

5.4.2. Model validation

Our experimental results have shown that the F1-scores of the DL models for network traffic, system logs, and host statistics are 0.93, 1.0, and 1.0, respectively. This raises the question of whether the outstanding performance might be attributed to overfitting. To further validate the model, we employed K-fold cross-validation on three different data sources. This technique involves dividing the dataset into K equally sized subsets and then training the model K times. Each time, one of the subsets serves as the validation set, while the remaining K-1 subsets form the training set. Finally, the performance evaluations from K validations are averaged to obtain an overall performance assessment of the model. The cross-validation F1-scores for network traffic, system logs, and host statistics are 0.913, 0.979, and 0.99, respectively, showing little deviation from the original results. This indicates that our model does not experience significant overfitting problems.

Table 12
Attack detection rate of the present method for different data sources.

	Mirai	Mining	Ransomware	Disk wipe	Endpoint DoS	Rootkit ransomware	Data theft	Initial access	C&C	Impact	Overall
All	11/11	30/30	33/33	36/36	64/64	37/37	30/30	43/43	20/20	178/178	241/241
Traffic+Statistics	11/11	30/30	33/33	36/36	64/64	37/37	30/30	43/43	20/20	178/178	241/241
Logs+Statistic	11/11	30/30	33/33	36/36	64/64	36/37	30/30	43/43	19/20	178/178	240/241
Logs+Traffic	7/11	4/30	5/33	5/36	10/64	12/37	6/30	13/43	12/20	24/178	49/241
Statistic	11/11	30/30	33/33	36/36	64/64	36/37	30/30	43/43	19/20	178/178	240/241
Traffic	7/11	4/30	4/33	4/36	4/64	12/37	5/30	11/43	12/20	17/178	40/241
Logs	3/11	1/30	1/33	1/36	7/64	1/37	1/30	2/43	2/20	11/178	15/241

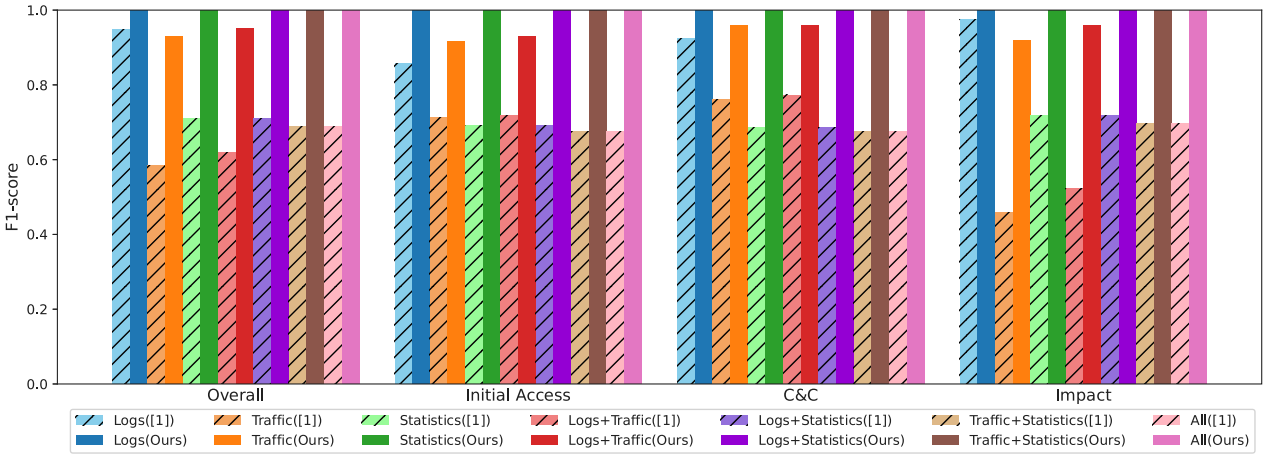


Fig. 4. Stage-based comparison of F1-scores obtained by the present method and that in [10].

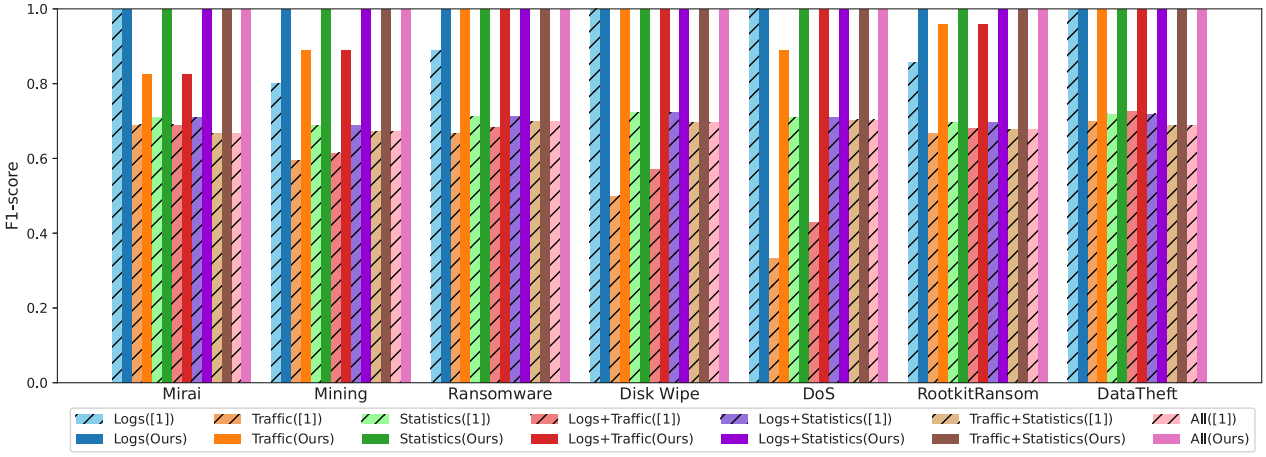
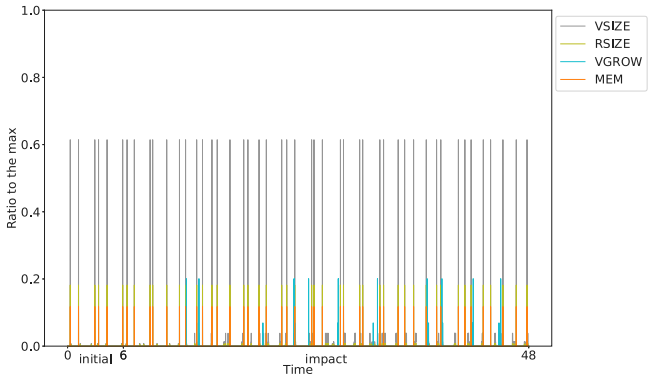


Fig. 5. Attack-based comparison of F1-scores obtained by the present method and that in [10].

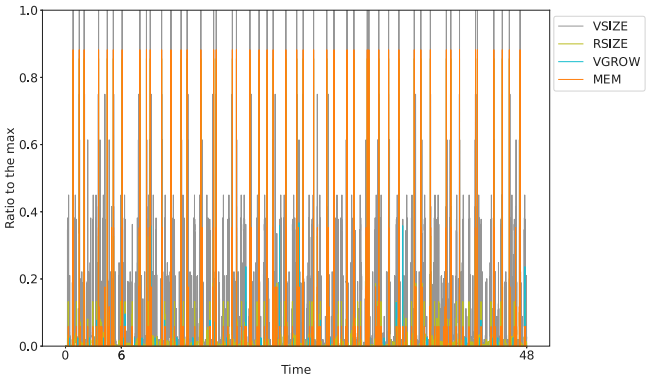
6. Conclusion

This study has proposed a DL approach for host-based IDS based on three different data sources, namely network traffic, system logs, and host statistics. The feasibility of the proposed approach has been demonstrated for seven different attack methods and three different attack stages (initial access, C&C command, and impact). In the proposed approach, the three datasets are collected from the CR»ME toolchain and preprocessed, and are then divided into a training set and testing set in a ratio of 80%:20% in accordance with their timestamps. The three datasets are handled by different DL models depending on their particular characteristics. In particular, the network traffic data are processed by a CNN to extract the data features automatically, the system logs data are processed by LSTM and a self-attention model to learn the temporal relationships, and the host statistics are processed by a DNN. The predictions of the three models for the host status (benign or malevolent) are then combined via an ensemble method to produce a final prediction for the host status. The prediction performance of the proposed method has been evaluated by means of the F1-score for seven combinations of the data sources (namely, three individual data sources, three combinations of two data sources, and one combination of all three data sources).

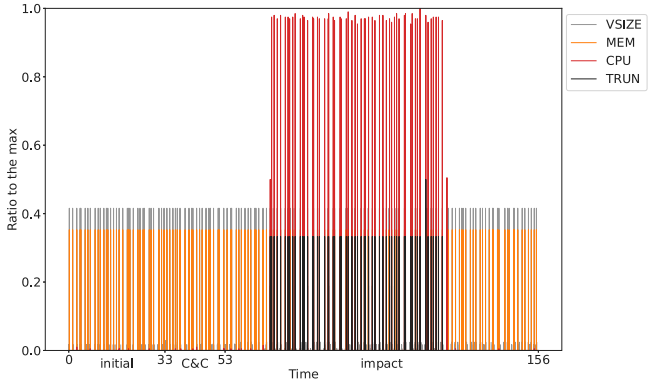
The experimental findings reveal that the F1-scores achieved by the DL models for network traffic, system logs, and host statistics are 0.93, 1.0, and 1.0, respectively. Although the F1-score of the system logs DL model reaches 1.0, the data source generates relatively few malicious data points. As a result, many of the time slots have no data points, and therefore the final prediction outcome is dependent on the prediction results of the other models. To address this problem, it is desirable to obtain the final prediction outcome for each time slot using an ensemble method based on the prediction outputs of all three data sources.



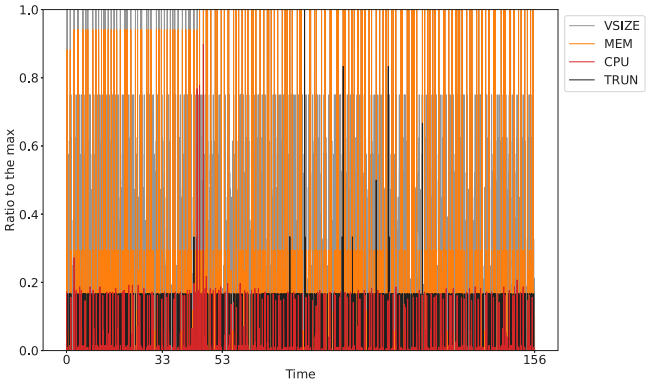
(a) Features of Mirai malicious dataset in each time slot



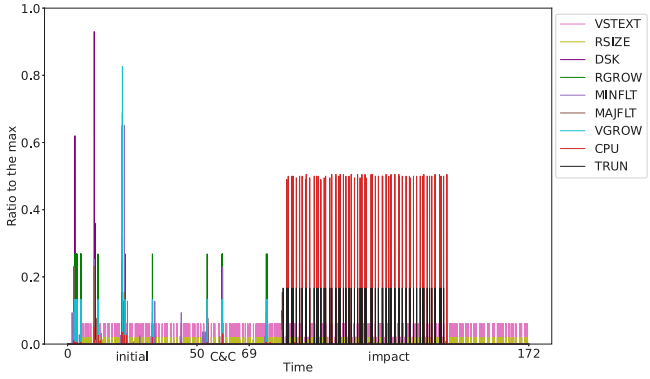
(b) Features of Mirai benign dataset in each time slot



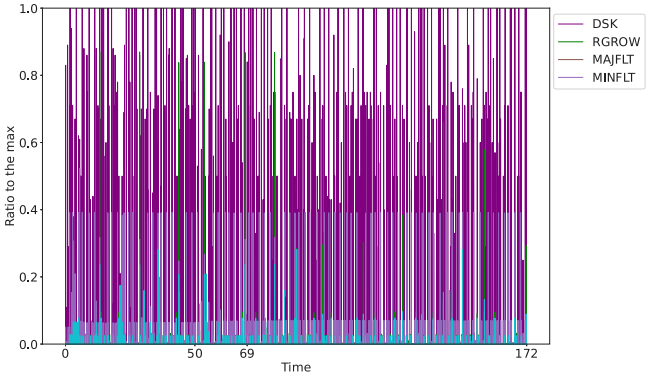
(c) Features of Mining malicious dataset in each time slot



(d) Features of Mining benign dataset in each time slot



(e) Features of Ransomware malicious dataset in each time slot



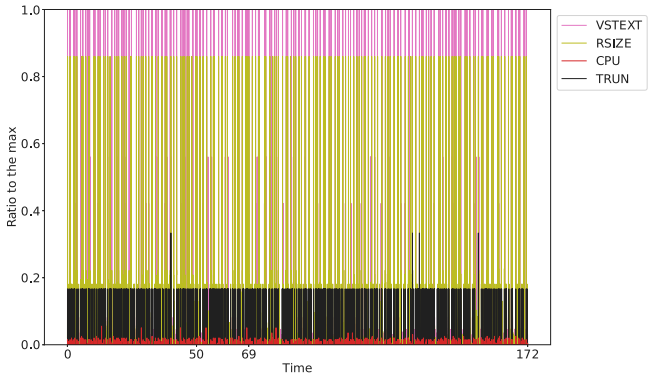
(f) Features of Ransomware benign dataset in each time slot (part 1)

Fig. A.1. Comparison of host statistic features impacted by different attack methods.

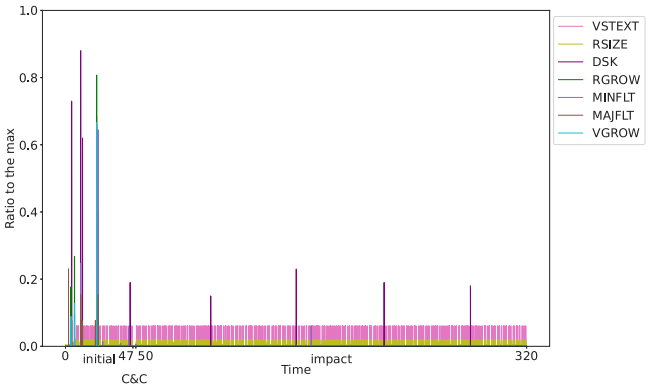
The experimental results have confirmed that the corresponding F1-score reaches 1.0 for all of the considered attack methods and attack stages. Furthermore, it has been shown that the proposed ensemble method significantly outperforms the XGBoost model presented in [10].

The experimental results have shown that, among the three data sources, the host statistics data source generates the largest number of malicious data points and provides the best detection capability. To investigate this finding further, a detailed analysis has been performed to examine the system resources primarily affected under different attack methods and attack stages. It has been shown that in the initial access and C&C attack stages, most of the attack methods increase the system resources related to page faults and memory, including DSK, MINFLT, MAJFLT, VGROW, and RGROW. However, in the impact stage, the affected resources depend on the particular attack method. For example, the Mining and Ransomware attacks affect the TRUN and CPU resources, while the Disk Wipe attack affects the DSK, VSIZE, RSIZE, and MEM resources, and the Data Theft attack affects the RDDSK and DSK resources.

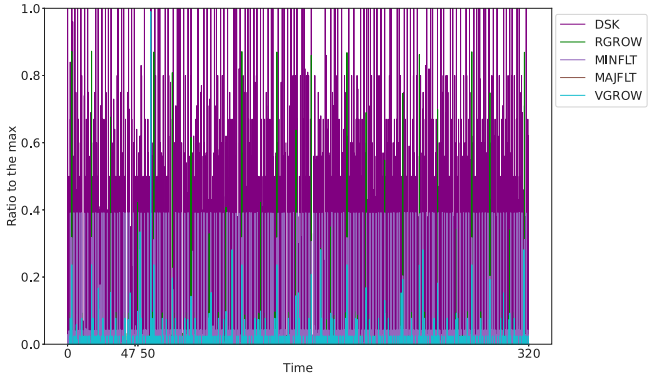
While the experimental results are promising, some limitations still exist. Our model prioritizes detection accuracy over resource consumption, which may not be suitable for IoT or mobile devices with limited computational resources. In the future, it may be possible to integrate edge servers, similar to [27], to reduce the resource demands on detection devices while simultaneously enabling the detection of multiple data sources.



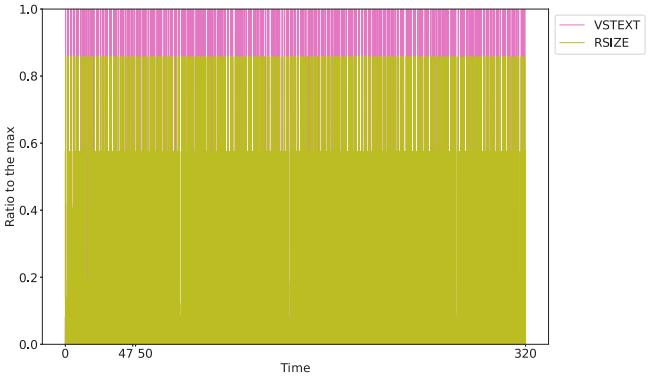
(g) Features of Ransomware benign dataset in each time slot (part 2)



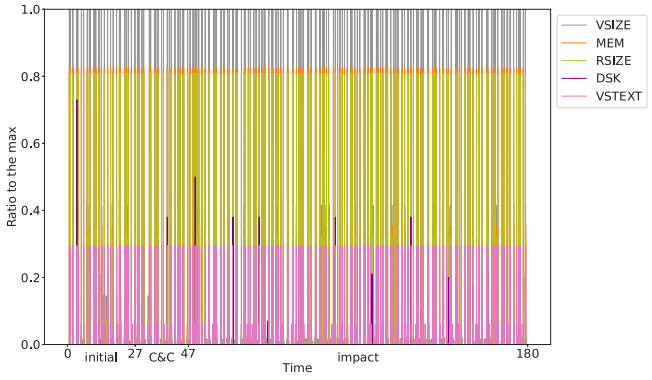
(h) Features of DoS malicious dataset in each time slot



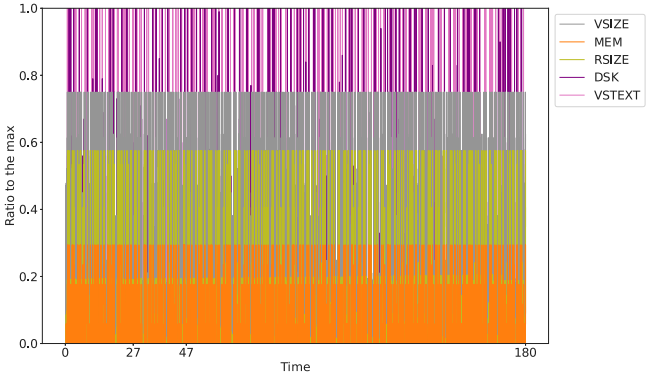
(i) Features of DoS benign dataset in each time slot (part 1)



(j) Features of DoS benign dataset in each time slot (part 2)



(k) Features of DiskWipe malicious dataset in each time slot



(l) Features of DiskWipe benign dataset in each time slot

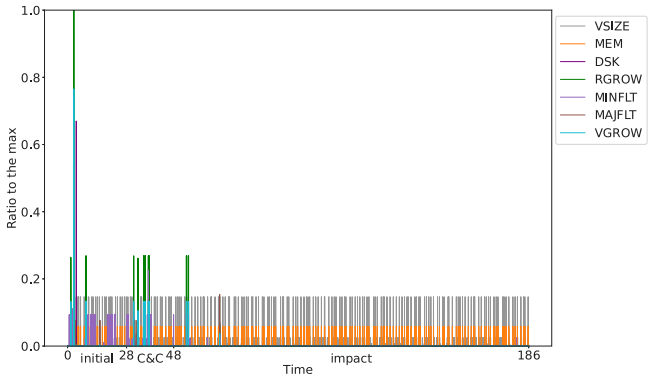
Fig. A.1. (*continued*).

Another important feature of an IDS is its capability to detect unknown attacks or zero-day attacks, which is not considered in this study. This aspect will be another focus of our future research.

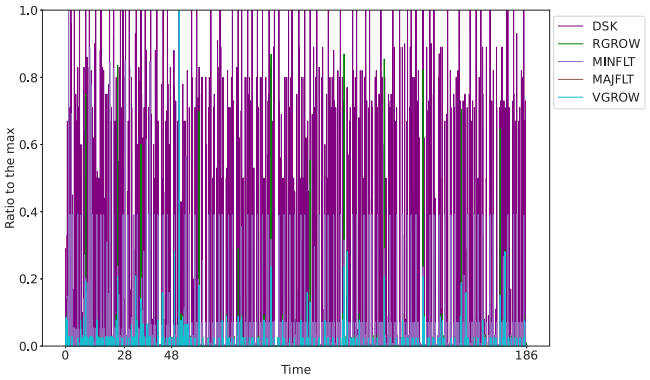
In addition, in our future studies, we will investigate the applicability of the proposed framework for intrusion detection in real-world environments. The data used in the present study (collected by CR»ME) is emulated data and may not accurately represent the various situations encountered in real-world networks (e.g., normal behavior that resembles attack-like behavior). Thus, future studies will also aim to increase the diversity and volume of the data used for training purposes, in order to obtain more robust models for detecting complex attacks.

CRediT authorship contribution statement

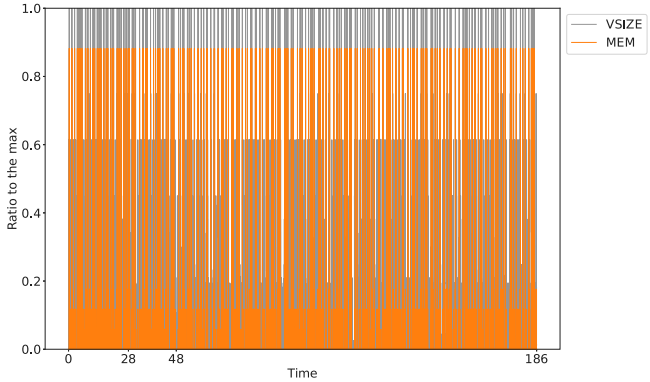
Ren-Hung Hwang: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Roles/Writing – original draft, Roles/Writing – review & editing. **Chieh-Lun Lee:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Visualization, Roles/Writing – original draft. **Ying-Dar Lin:** Conceptualization, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Roles/Writing – review & editing. **Po-Chin Lin:** Conceptualization, Investigation, Methodology, Supervision, Roles/Writing – review & editing. **Hsiao-Kuang Wu:**



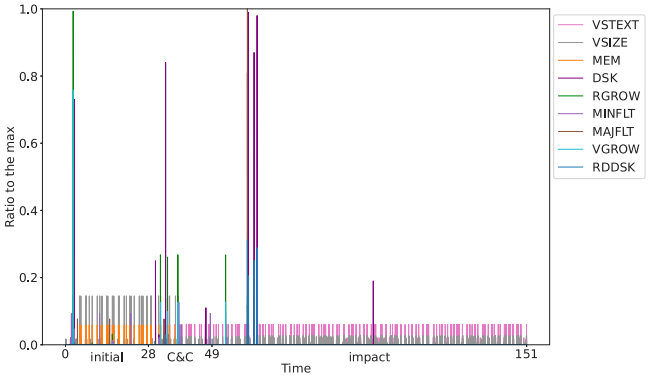
(m) Features of Rootkit Ransomware malicious dataset in each time slot



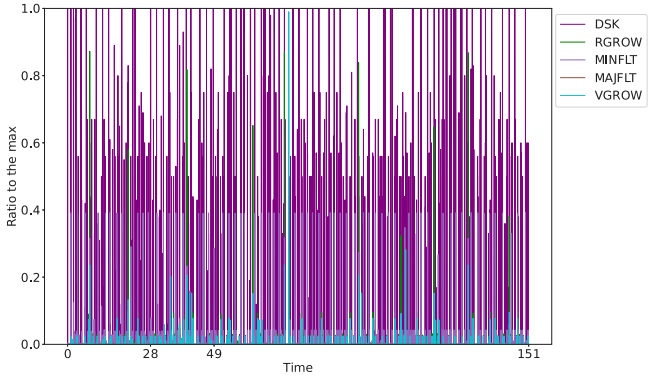
(n) Features of Rootkit Ransomware benign dataset in each time slot (part 1)



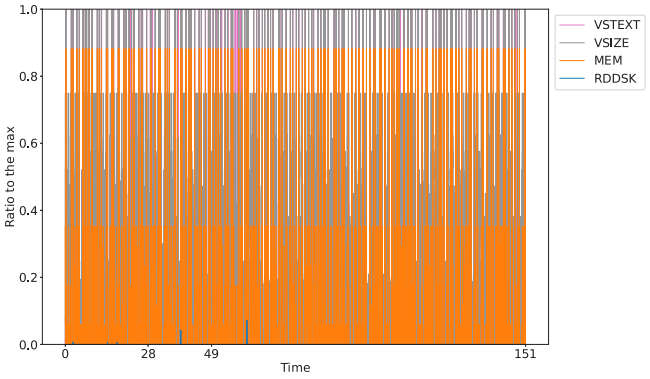
(o) Features of Rootkit Ransomware benign dataset in each time slot (part 2)



(p) Features of DataTheft malicious dataset in each time slot



(q) Features of DataTheft benign dataset in each time slot (part 1)



(r) Features of DataTheft benign dataset in each time slot (part 2)

Fig. A.1. (*continued*).

Investigation, Methodology, Supervision, Roles/Writing – review & editing. **Yuan-Cheng Lai:** Investigation, Methodology, Supervision, Roles/Writing – review & editing. **C.K. Chen:** Funding acquisition, Investigation, Supervision, Roles/Writing – review & editing.

Declaration of competing interest

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

Data availability

The data is generated by an open source tool. Data will be made available upon request or readers can generated by themselves by the open source tool.

Acknowledgments

This research work was partially supported by the National Science and Technology Council (NSTC), Taiwan, under the grant number NSTC 111-2221-E-A49-193-MY3.

Table A.1
Attack behaviors of different attack methods.

Attack scenario	Initial access	C&C	Impact
Mirai	T1190 Exploit Public-Facing Application	T1105 Ingress Tool Transfer	T1498 Network Denial of Service
Mining	T1190 Exploit Public-Facing Application	T1008 Fallback Channels	T1496 Resource Hijacking
Ransomware	T1078 Valid Accounts	T1008 Fallback Channels	T1486 Data Encrypted for Impact
DiskWipe	T1078 Valid Accounts	T1008 Fallback Channels	T1561 Disk Wipe
Endpoint DoS	T1078 Valid Accounts	Create accounts	T1499 Endpoint Denial of Service
Rootkit Ransomware	T1078 Valid Accounts	T1008 Fallback Channels	T1486 Data Encrypted for Impact
DataTheft	T1078 Valid Accounts	T1008 Fallback Channels	Data theft

Appendix. Feature analysis of host statistics

The overall objective of the investigation was to determine which system resources had the greatest impact on the model prediction. Since the value of each system resource feature varies widely, all of the collected data were normalized to the interval of 0 and 1 by dividing the collected value by the maximum value of the corresponding resource. Each attack was divided into three stages and thus showed a different system resource usage pattern in each stage. Note that the ATT&CK technique used by each attack at each stage is shown in Table A.1.

Fig. A.1 shows the main features that were found to be significantly different between the malicious dataset and benign dataset for each of the seven attack scenarios. For each figure, the x-axis shows the proportion of system resources used by the attack processes in each time slot. Note that the blank spaces indicate the absence of any benign or malicious processes in the corresponding slot. Note also that for each attack, the graph on the left shows the usage status of the system resources that are mainly affected by the attack, while the graphs in the middle and on the right, respectively, show the usage status of the system resources in the corresponding benign dataset. Overall, the results show that each of the attack modes has certain host statistics features which differ strongly between the malicious and benign processes, and which can therefore be learned by the DL model to detect the network status (benign or malicious) in each time slot.

A further detailed examination was made of the main features affected by each attack at the different attack stages. In the initial access attack stage, Mirai scans and cracks devices with weak telnet passwords. Similarly, Mining exploits vulnerabilities in the Apache Continuum service to obtain access to the system. Ransomware and End-point DoS exploit vulnerabilities in the IRC server and docker daemon to access the system and obtain privilege escalation. Meanwhile, Disk Wipe, Rootkit Ransomware, and Data Theft all use the Ruby on Rails web application vulnerability to gain root privileges. In the C&C attack stage, Mirai downloads and executes malicious binary code. Endpoint DoS creates accounts on the compromised device, and the other five attacks insert a backdoor into the compromised machine. The initial access and C&C stages exhibit similar system resource usage patterns for these attacks, including DSK, MINFLT, MAJFLT, VGROW, and RGROW. For example, some attack files cannot be found in memory, which causes the system to suffer page faults and grab the files from disk to memory. Consequently, the features related to page faults, such as DSK, MINFLT, and MAJFLT, all increase. The two other features, i.e., VGROW and RGROW, representing the growth sizes of the virtual memory and resident memory, respectively, similarly increase. Fig. A.1 also shows that the usage of the system resources affected by the same attack technique is similar. For example, when obtaining root privileges using the Ruby on Rails web application vulnerability, the RGROW feature increases to 1 in the initial attack stage, and a significant increase in VGROW and RGROW occurs when inserting a backdoor (see Figs. A.1(m) and (p)).

In the impact attack stage, the different attacks affect different system resources. For example, Mining hijacks the system resources to mine Monero currency, while Ransomware encrypts files and hence consumes a large amount of CPU resources. Thus, the TRUN and CPU usage features increase significantly. Disk Wipe deletes the essential or sensitive files on the device, together with the affected directories, and hence affects the disk and memory-related features, such as DSK, VSIZE, RSIZE, and MEM. Data Theft reads and steals a large amount of data from the disk, and thus RDDSK and DSK both increase. Rootkit Ransomware, however, shows no obvious effect on the system resources during the impact stage since it hides the attack behavior. In general, some system resource usage statistics are common for benign processes, but are very different for attack processes, including VSTEXT, VSIZE, and RSIZE.

References

[1] Jose S, Malathi D, Reddy B, Jayaseeli D. A survey on anomaly based host intrusion detection system. J Phys Conf Ser 2018;1000:012049.

[2] Bridges R, Glass-Vanderlan T, Iannacone M, Vincent M, Chen Q. A survey of intrusion detection systems leveraging host data. ACM Comput Surv 2019;52:1–35.

[3] Aldweesh A, Derhab A, Emam AZ. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. Knowl-Based Syst 2020;189:105124.

[4] Horng S-J, Su M-Y, Chen Y-H, Kao T-W, Chen R-J, Lai J-L, et al. A novel intrusion detection system based on hierarchical clustering and support vector machines. Expert Syst Appl 2011;38(1):306–13.

[5] Aburomman AA, Ibne Reaz MB. A novel SVM-kNN-PSO ensemble method for intrusion detection system. Appl Soft Comput 2016;38:360–72.

[6] Strom BE, Applebaum A, Miller DP, Nickels KC, Pennington AG, Thomas CB. Mitre att&ck: Design and philosophy. Technical report, The MITRE Corporation; 2018.

[7] Milajerdi SM, Gjomemo R, Eshete B, Sekar R, Venkatakrishnan V. HOLMES: Real-time APT detection through correlation of suspicious information flows. In: 2019 IEEE symposium on security and privacy. 2019, p. 1137–52.

[8] Ghafir I, Hammoudeh M, Prenosil V, Han L, Hegarty R, Rabie K, et al. Detection of advanced persistent threat using machine-learning correlation analysis. Future Gener Comput Syst 2018;89:349–59.

[9] Bodström T, Hämäläinen T. A novel deep learning stack for APT detection. Appl Sci 2019;9(6).

[10] Lin Y-D, Wang Z-Y, Lin P-C, Nguyen V-L, Hwang R-H, Lai Y-C. Multi-datasource machine learning in intrusion detection: Packet flows, system logs and host statistics. J Inf Secur Appl 2022;68:103248.

[11] Li X, Chen W, Zhang Q, Wu L. Building auto-encoder intrusion detection system based on random forest feature selection. Comput Secur 2020;95:101851.

[12] Hwang R-H, Peng M-C, Huang C-W, Lin P-C, Nguyen V-L. An unsupervised deep learning model for early network traffic anomaly detection. IEEE Access 2020;8:30387–99.

[13] Du M, Li F, Zheng G, Srikumar V. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. New York, NY, USA: Association for Computing Machinery; 2017, p. 1285–98.

[14] Meng W, Liu Y, Zhu Y, Zhang S, Pei D, Liu Y, et al. LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: Proceedings of the twenty-eighth international joint conference on artificial intelligence. International Joint Conferences on Artificial Intelligence Organization; 2019, p. 4739–45.

[15] Ribeiro J, Saghezchi FB, Mantas G, Rodriguez J, Abd-Alhameed RA. HIDROID: Prototyping a behavioral host-based intrusion detection and prevention system for android. IEEE Access 2020;8:23154–68.

[16] Ham H-S, Kim H-H, Kim M-S, Choi M-J. Linear SVM-based android malware detection for reliable IoT services. J Appl Math 2014;2014:594501.

[17] Ribeiro J, Saghezchi FB, Mantas G, Rodriguez J, Shepherd SJ, Abd-Alhameed RA. An autonomous host-based intrusion detection system for android mobile devices. Mob Netw Appl 2020;25(1):164–72.

[18] Zhang X, Ran J, Mi J. An intrusion detection system based on convolutional neural network for imbalanced network traffic. In: 2019 IEEE 7th international conference on computer science and network technology. 2019, p. 456–60.

[19] Zeng Y, Gu H, Wei W, Guo Y. Deep-full-range : A deep learning based network encrypted traffic classification and intrusion detection framework. IEEE Access 2019;7:45182–90.

[20] Sun P, Liu P, Li Q, Liu C, Lu X, Hao R, et al. DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system. Secur Commun Netw 2020;2020:8890306.
[21] Hwang R-H, Peng M-C, Nguyen V-L, Chang Y-L. An LSTM-based deep learning approach for classifying malicious traffic at the packet level. Appl Sci 2019;9(16).
[22] Zhang X, Xu Y, Lin Q, Qiao B, Zhang H, Dang Y, et al. Robust log-based anomaly detection on unstable log data. In: Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering. ESEC/FSE 2019, New York, NY, USA: Association for Computing Machinery; 2019, p. 807–17.
[23] Tan Z, Pan P. Network fault prediction based on CNN-LSTM hybrid neural network. In: 2019 International conference on communications, information system and computer engineering. 2019, p. 486–90.
[24] Wang X, Cao Q, Wang Q, Cao Z, Zhang X, Wang P. Robust log anomaly detection based on contrastive learning and multi-scale MASS. J Supercomput 2022.
[25] Yang R, Qu D, Gao Y, Qian Y, Tang Y. nLSALog: An anomaly detection framework for log sequence in security management. IEEE Access 2019;7:181152–64.
[26] Sun C-C, Sebastian Cardenas DJ, Hahn A, Liu C-C. Intrusion detection for cybersecurity of smart meters. IEEE Trans Smart Grid 2021;12(1):612–22.
[27] Mudgerikar A, Sharma P, Bertino E. E-Spion: A system-level intrusion detection system for IoT devices. In: Proceedings of the 2019 ACM asia conference on computer and communications security. New York, NY, USA: Association for Computing Machinery; 2019, p. 493–500.
[28] Bui H-K, Lin Y-D, Hwang R-H, Lin P-C, Nguyen V-L, Lai Y-C. CrÉme: A toolchain of automatic dataset collection for machine learning in intrusion detection. J Netw Comput Appl 2021;193:103212.
[29] He P, Zhu J, Zheng Z, Lyu MR. Drain: An online log parsing approach with fixed depth tree. In: 2017 IEEE international conference on web services. 2017, p. 33–40.
[30] Chen T, Chen Y, Lv M, He G, Zhu T, Wang T, et al. A payload based malicious HTTP traffic detection method using transfer semi-supervised learning. Appl Sci 2021;11(16).
[31] Khan A, Sohail A, Zahoora U, Qureshi AS. A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 2020;53(8):5455–516.
[32] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Advances in neural information processing systems, vol. 30. Curran Associates, Inc.; 2017.