Received 5 November 2020; revised 20 November 2020; accepted 7 December 2020. Date of publication 16 December 2020; date of current version 14 January 2021.

Digital Object Identifier 10.1109/OJCOMS.2020.3044323

Adaptive Intrusion Detection in the Networking of Large-Scale LANs With Segmented Federated Learning

YUWEI SUN (Student Member, IEEE), HIROSHI ESAKI (Member, IEEE), AND HIDEYA OCHIAI (Member, IEEE)

Graduate School of Information Science and Technology, University of Tokyo, Tokyo 1138654, Japan

CORRESPONDING AUTHOR: Y. SUN (e-mail: sywtokyo@hongo.wide.ad.jp)

ABSTRACT Predominant network intrusion detection systems (NIDS) aim to identify malicious traffic patterns based on a handcrafted dataset of rules. Recently, the application of machine learning in NIDS helps alleviate the enormous effort of human observation. Federated learning (FL) is a collaborative learning scheme concerning distributed data. Instead of sharing raw data, it allows a participant to share only a trained local model. Despite the success of existing FL solutions, in NIDS, a network's traffic data distribution does not always fit into the single global model of FL; some networks have similarities with each other but other networks do not. We propose Segmented-Federated Learning (Segmented-FL), where by employing periodic local model evaluation and network segmentation, we aim to bring similar network environments to the same group. A comparison between FL and our method was conducted against a range of metrics including the weighted precision, recall, and F1 score, using a collected dataset from 20 massively distributed networks within 60 days. By studying the optimized hyperparameters of Segmented-FL and employing three evaluation methods, it shows that Segmented-FL has better performance in all three types of intrusion detection tasks, achieving validation weighted F1 scores of 0.964, 0.803, and 0.912 with Method A, Method B, and Method C respectively. For each method, this scheme shows a gain of 0.1%, 4.0% and 1.1% in performance compared with FL.

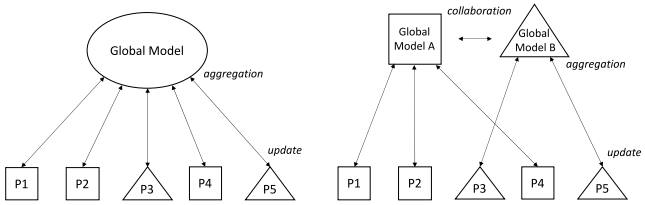
INDEX TERMS Cybersecurity, deep learning, intrusion detection, segmented-federated learning, LAN, convolutional neural network.

I. INTRODUCTION

NETWORK intrusion detection strategies existing in current systems have been revealing issues of low adaptivity to network traffic from various network environments. A network monitoring system that has the ability to detect and track malware in various networks is highly demanded. The scheme of federated learning (FL) was first proposed by Google to solve problems of data scarcity and privacy in the field of machine learning [1]. This scheme has been employed in applications such as image recognition, natural language processing, cybersecurity, and so on. It showed that by using this scheme, users could share intelligence on a machine learning task with each other, whereas without disclosing their raw data.

Despite the success of existing FL solutions [2], [3], in NIDS, a network's traffic data does not always fit into the single global model of FL; some networks have similarities with each other but other networks do not. In this article, we propose a novel adaptive learning scheme of Segmented-Federated Learning (Segmented-FL) (Fig. 1). We employ periodic local model evaluation and segmentation for adaptive model training. Different from all users training a model under the single global model at the central server in FL, Segmented - FL has a feature that each segmented group of users is arranged with a specified global model for adaptive learning.

The Segmented - FL is utilized for parameter sharing among participants as well as automatic segmentation of participants thus adapting to massively distributed networks. For each day, also called round, selected participants conduct local model training based on locally collected datasets. Then the trained models are uploaded and aggregated in the



(a) Federated Learning (b) Segmented-Federated Learning

FIGURE 1. Segmented-FL has multiple global models, each of which aggregates local models of similar participants. E.g. since P1, P2, and P4 are similar; P3 and P5 are similar, they are segmented into two groups belonging to Global Model A and Global Model B respectively.

parameter server for updating a global model. The periodic evaluation of local models is used to validate and divide them into several groups for adaptive learning based on recent performance in tasks of intrusion detection. If a model shows a disadvantaged performance compared with the other models, it will be removed from the current group, and a new group with an independent global model will be initialized. Thus, from the next round, these participants will train their local models using respective global models.

A deep convolutional neural network (CNN) is used as the protocol for model training and sharing. We adopt a combined approach of feature representation and expert knowledge-based labeling of benign and malicious network events to train an artificial neural network (ANN) model. The trained models are used to detect malicious network events in LANs, and enhance detection performance through ANN model parameters sharing among group members.

The patterns of network traffic data for intrusion detection are varying in large-scale network environments, with various network scales, devices with different operating systems and applications installed, and so on. For example, some LANs have more than 1000 active users while some have only a dozen users, showing different presence information. Devices in these LANs are working on different operating systems such as Windows, macOS, and Ubuntu, which reveal different traffic patterns. Some LANs include the Internet of Things (IoT) devices such as Web cameras and smart home devices. Besides, the user groups could be different from students in academic institutes to professionals in industries. These reasons above contribute to the diversity of network traffic patterns as well as the difficulty of intrusion detection in large-scale distributed networks.

The research project of the LAN-Security Monitoring Project aims to collect these varying real-time network traffic data from massively distributed academic networks [4]. In this project, a monitoring device was developed and deployed in a network to collect traffic data. The collected data is saved at a server for further analysis, including complete broadcast traffic data in a local area network (LAN) and network traffic sent directly to monitoring devices. A total of 38 institutes have been participating with a monitoring device set up in an office network or laboratory network. The data from this project provided the basis of datasets used in the research of Segmented-FL.

The challenges and contributions of the research include:

- For segmenting similar participants into the same group with a respective global model, we propose Segmented-FL for adaptive model learning and sharing.
- To understand the effect of key parameters of Segmented-FL, we have conducted a comprehensive survey on 27 combinations of them for exploiting the optimized solution.
- For validating the model with diverse network traffic data, we applied massively distributed networks' real-time traffic from various academic institutes.
- To compare the performance between FL and our method, we have conducted evaluations against a range of metrics, weighted precision, recall, and F1 score. It shows that our method has better performance in intrusion detection with large-scale distributed networks.

This article is organized as follows. Section II discusses related works on network intrusion detection and the application of federated learning in cybersecurity. Section III provides an overview of our method consisting of fundamental federated learning, intrusion detection with a combined approach of visual analytics and CNN, and the proposed Segmented-FL. Section III presents experiments and performance evaluation against the metrics of weighted precision, recall, and F1 score. Section V discusses the pros and cons of this method. Section VI concludes this article.

II. RELATED WORK

Traditional approaches to network intrusion detection are commonly related to matching between the database of known malicious patterns based on expert knowledge and network behavior in a network [5], [6], [7]. Many NIDSs [8], [9] use a network-based activity study to determine if a node is compromised, such as traffic or frequency analysis, and deep packet inspection. Unfortunately, these approaches appear to be insufficient, with underlying issues of adaptivity and privacy.

Recently, machine learning and neural networks have been employed to strengthen the performance of network systems for intrusion detection in personal computers and critical infrastructures, such as support vector machine (SVM) and artificial neural network (ANN) [10], [11]. Besides, due to the rapid development of deep learning in recent years, data analysis on big data of network traffic became feasible and was employed in lots of research. For instance, Salama *et al.* [12] presented an intrusion detection hybrid scheme using deep belief network and SVM, recognizing the malicious network events. An evaluation based on the NSL-KDD dataset was conducted, with a detection accuracy of above 0.9. Yang *et al.* [13] adopted a combined approach of using restricted Boltzmann machine to extract high-level feature representation of network traffic and classifying these features with SVM. Duy and Diep [14] presented intrusion detection based on the NSL-KDD dataset,

using a feedforward neural network. Their model achieved an F1 score of 0.962 at last. Saxe and Berlin [15] proposed a deep neural network-based malware detector by employing two-dimensional binary program features. Yousefi-Azar *et al.* [16] presented a generative feature learning-based approach for malware classification, where they extracted latent features of network traffic based on an unsupervised learning model called autoencoder.

The aforementioned research on intrusion detection aims to detect malware in limited network environments. Detection strategies and results from one network usually could not be applied to other network environments. Federated learning (FL) is a collaborative learning scheme that allows users to share intelligence on machine learning tasks with each other thus improving overall performance. Different from centralized learning [17], FL allows users to share local models instead of raw network traffic data, which might reveal private information to the learning group. There have been already lots of research on the application of FL in cybersecurity and edge computing. For example, Abeshu and Chilamkurti [2] proposed a cyberattack detection model using FL to improve accuracy in detecting attacks. It showed enhanced performance and privacy of participants as well as reduced traffic load. Wang *et al.* [3] presented an intrusion detection in simulated environments based on FL, with a gradient descent-based learning function. Schneible and Lu [18] demonstrated an application of FL in anomaly detection, where autoencoder was employed for analytics and observations. Zhao *et al.* [19] employed FL to tackle the data scarcity problem and to preserve data privacy, where multiple participants collaboratively train a global model. Daga *et al.* [20] proposed Cartel, a system for collaborative learning in edge clouds. The CICIDS2017 [21] Intrusion Detection evaluation dataset was used, consisting of benign data samples, distributed denial-of-service (DDoS) attacks, and port scan attacks. Additionally, Yu *et al.* [22] presented a Mobility-aware Proactive edge Caching scheme based on Federated learning (MPCF) for predicting content popularity with the private training data distributed on local vehicles. This scheme could adapt to various mobility patterns and references of vehicles and protect users' privacy.

Different from former research, we propose Segmented-FL for adaptive intrusion detection in large-scale massively distributed networks. The proposed approach has a feature that the scheme architecture is adjusted continually based on periodic evaluation results of local models thus adapting to varying traffic data in these network environments. A deep CNN model is employed as the protocol ANN for collaborative learning, using feature representation of network traffic as the input, results from knowledge-based labeling as the ground truth.

III. ADAPTIVE NETWORK INTRUSION DETECTION BASED ON SEGMENTED-FL

A. FEDERATED LEARNING

Federated learning is a collaborative learning scheme for users to share intelligence on model training with each other, without sharing their private data. The intrusion detection in networks could cause exposure of private information to third-party entities while analyzing network traffic data for tracking malicious users. The implementation of FL in intrusion detection allows network operators to detect malware without accessing users' private data.

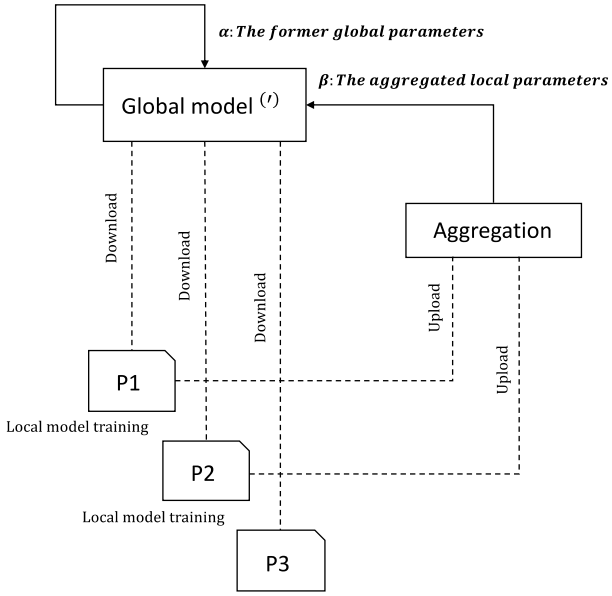


FIGURE 2. The architecture and mechanism of federated learning.

The architecture and mechanism of FL for collaborative learning by sharing local model training results is shown below (Fig. 2).

In this graph, first, participants of FL download the latest global model from the central server and update their local models. Then, the selected users will conduct model training based on a local dataset with varying numbers of data and portions of different clusters. After training, these trained local models will be uploaded to the server for aggregation. As a result, based on the aggregated local parameters and the former global parameters, the current global model is updated for the next round's collaborative learning.

B. INTRUSION DETECTION WITH A COMBINED APPROACH OF VISUAL ANALYTICS AND A CONVOLUTIONAL NEURAL NETWORK

To process network traffic for model training, we generated feature maps representing communication patterns of various network protocols by employing a quantitative approach based on frequency information. These extracted information of protocols includes IP, ARP, TCP, HTTP, HTTPS, UDP, mDNS, DHCP, and Others. Fineness, a time window parameter, which corresponds to generating a pixel value in a feature map, was employed when extracting protocol information from raw data. By studying varying fineness rates for the efficiency of feature representation [23], we found a larger time window would reveal more patterns of network traffic in the feature map, while a smaller one would contribute to

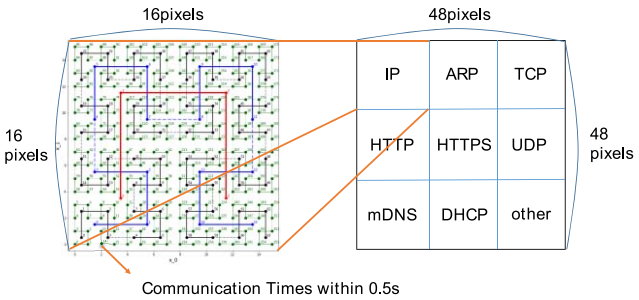


FIGURE 3. Protocol-wise feature representation of network traffic in a LAN based on the Hilbert curve and array exchange.

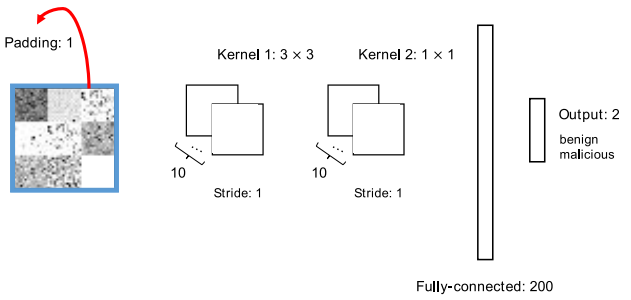


FIGURE 4. The architecture of the four-layers CNN model.

the sensibility of the detection systems. Given the balance between the pattern representation ability and the detection sensibility, we selected 0.5 seconds as fineness for computing frequencies, recording how many packets related with a specified protocol were sent or received within the defined time window, thus generating a feature map.

Then a chunk of these records with a size of 256 for each protocol was combined to generate communication patterns related with these protocols. As a result, each generated communication pattern covers network traffic features within 128 seconds, related with a specified protocol. The reason for choosing a size of 256 is that the selected time window for generating a pattern allows a network operator to respond to these detected malicious behaviors timely. The relationship between the time window for pattern generation and the fineness rate is defined in (1).

$$T = T_{st} \cdot fineness \cdot s^2 \tag{1}$$

where T represents the time window for pattern generation, *Tst* (Time Standard) represents the standard interval of one second for generating a pattern, fineness represents the time window for record generation thus controlling information density in a pattern, and s represents the pattern size (side length).

To combine every 256 records into a pattern, we converted frequency information into pixel values by employing (2). Then the Hilbert curve was used to generate a feature map, which is a geometric structure used to transform the structure of data so that it fills up all space in an image, by projecting pixel information to specified positions in the Hilbert curve.

$$p_i = \frac{c_i}{Max(c)} \cdot 255 \tag{2}$$

where p_i is pixel information, c_i is frequency information, and c represents all frequency information covered in a communication pattern. Thus, $\frac{c_i}{Max(c)}$ is in the interval of (0, 1], and p_i is in the interval of (0, 255].

By employing the array exchange, we aggregated these generated feature maps of nine types' protocols into one feature map representing protocol-wise network traffic features within 128 seconds in a LAN. Each generated feature map has a size of 48 pixels × 48 pixels (Fig. 3).

To process these feature maps of network events, we employed a supervised learning method of CNN (Fig. 4), since the generated feature maps are two-dimensional data with time-related information encoded. Considering possible training computation and transmitting time cost while model sharing, we employed a four-layers CNN which consists of two convolution layers each of which is followed by a maxpooling layer, and two fully-connected layers. For the first convolution layer, ten kernels which are used for convolution operation with a size of 3×3 and a stride of one were used. For the second convolution layer, ten kernels with a size of 1×1 and a stride of one were used. The fully-connected layer of hidden layers consists of 200 neurons, and the output layer consists of one neuron for classification between the benign and the malicious. The padding with a value of one pixel was used in computation at each convolution layer.

The learning function is used to update parameters of weights and biases in the model, with the purpose of decreasing prediction loss as much as possible. The RMSProp defined in (3) was employed for the model training, which has a characteristic that the emphasis is placed on the latest gradient information more than the past gradient information and gradually the past gradient information is forgotten, instead, the new gradient information is greatly reflected. The learning rate is used to control the step length for each epoch's update. Besides, the cross-entropy defined in (4) was employed as the loss function to compute prediction loss for updating.

$$\begin{aligned} \mathrm{h_t} &= \rho * \mathrm{h_{t-1}} + (1-\rho) * \frac{\partial \mathrm{L}}{\partial \mathrm{W_t}} \odot \frac{\partial \mathrm{L}}{\partial \mathrm{W_t}} \\ \mathrm{W_{t+1}} &= \mathrm{W_t} - \eta \frac{1}{\sqrt{\mathrm{h_t} + \epsilon}} \odot \frac{\partial \mathrm{L}}{\partial \mathrm{W_t}} \end{aligned} \tag{3}$$

where L represents the loss, W represents the weight, η is the learning rate, and ρ is a decay rate of RMSProp for controlling the forgetting rate of past gradient information which is usually set to a value of 0.9.

$$\mathrm{L} = -\sum_{\mathrm{k}}^{\mathrm{K}} \mathrm{t_k log y_k} \tag{4}$$

where K is the number of nodes at the output layer, k represents the kth node at the output layer, y_k is the prediction result, and t_k is the ground truth.

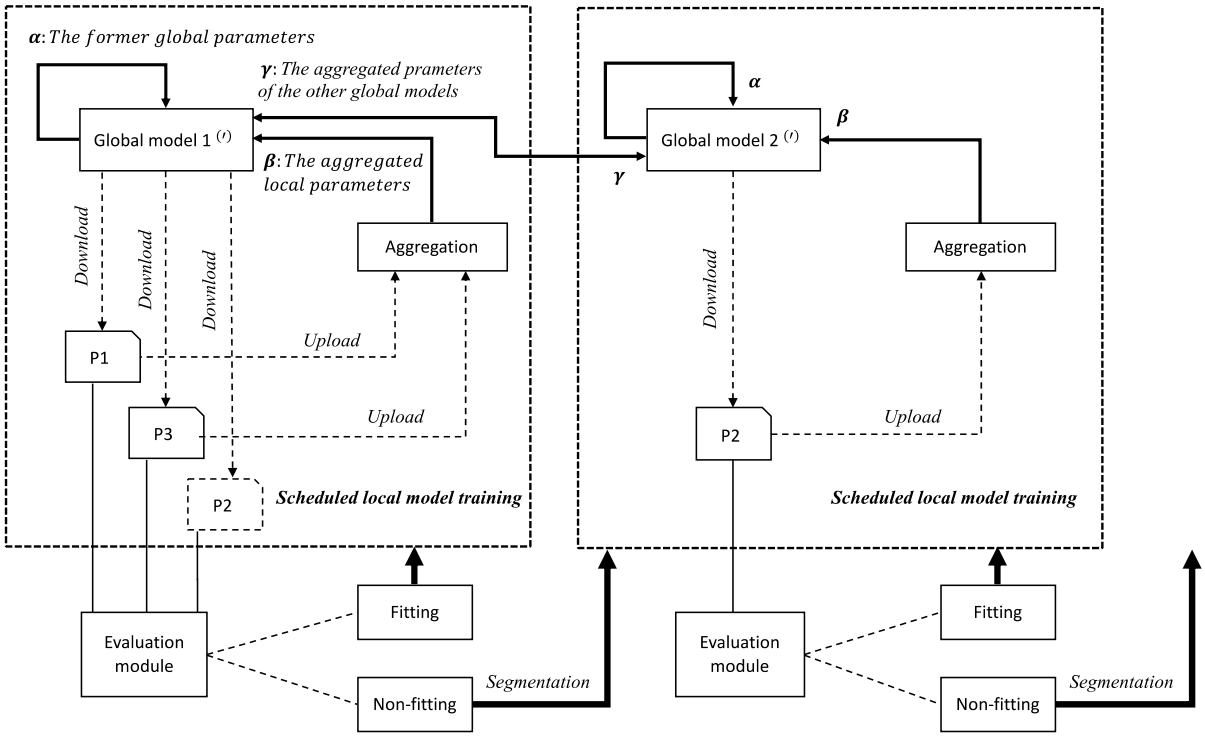


FIGURE 5. Periodical evaluation and segmentation: if a participant's model fits into the current global model according to the result of evaluation module, they remain in the group, but if the participant does not fit into the current global model, we move it to another group.

The Sigmoid function defined in (5) was used as the activation function at the last layer for narrowing the output into the range of (0, 1).

$$Sig(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

where x is the computation results of the last layer, and $Sig(x)$ is the output of the Sigmoid function with numerical values in the range of (0, 1).

C. SEGMENTED-FEDERATED LEARNING

Due to network environment diversity, a network's traffic data does not always fit into the single global model of FL; some networks have similarities with each other but others do not. We propose a novel adaptive learning scheme of Segmented-FL (Fig. 5), to safeguard the data privacy of participants as well as adapting to the various network environments by employing multiple global models.

In detail, the periodical performance evaluation is used for the quality verification of local model training in recent several rounds, based on the evaluation module defined in (6). By employing the average validation result as the metric, we periodically evaluate the performance of each participant's local model for training and sharing with the current global model. If a participant's model fits into the current global model according to the result of evaluation module, they remain in the group, but if the participant does not fit into the current global model, we move it to another group. If there is no group already created, a new group will be initialized based on the average-aggregated result of them. To decide whether a participant's model fits into the current global model, we apply a flexible threshold, and an evaluation result below the threshold represents it doesn't fit. Additionally, a relatively low threshold brings fewer participants for segmentation, whereas, a relatively high threshold brings more participants to a new global model. The participants moved into the new group conduct the next round's collaborative learning with the initialized global model. The maximum number of possibly existing global models can be set to five for the sake of simplicity.

$$\begin{aligned} d_i &= E_i - \frac{\sum_{i=1}^{n} E_i}{n} \\ e_i &= \frac{1}{1 + e^{-d_i}} \\ threshold &= 0.5 - h_f \times 0.01 \end{aligned} \tag{6}$$

where n is the number of participants, E_i represents the average validation result of participant i in the recent several rounds, d_i is the computation result of the difference between participant i's performance and the average performance of all group members, e_i is the output of the evaluation module using the Sigmoid function to covert d_i into the interval of (0, 1), and h_f is the segmentation fineness to adaptively adjust the threshold.

For the parameter aggregation to update a global model, we applied the former global parameters, the average-aggregated local parameters, and the average-aggregated

Algorithm 1 *SEGMENTED-FL*

$p_{g=1}$ is the initialized global model. N_g is the number of participants related to global model g. N_t is the number of participants conducting local model training in a group. D_t is the local dataset of participant t. G represents all global models. h_t is the number of rounds for periodic local model evaluation. L_g is a list including segmentation information of participants. B is the batch size. E is the local training epoch. η is the learning rate. α, β and γ are the component ratios for aggregation. p_{new} is the newly initialized global model from the segmentation.

Server executes:
  initialize $p_{g=1}$
  for each round $t = 1, 2, \ldots$ **do**
    for each global model g $= 1, 2, \ldots$ **do**
      $B_t \leftarrow$ (split N_g into batches with a size N_t)
      $s_t \leftarrow$ (participants conducting model training from B_t)
      $s_r \leftarrow$ (the other participants not conducting model training)
      for each participant $t \in s_t$ **in parallel do**
        Execute(p_t, D_t)
      for each participant $r \in s_r$ **in parallel do**
        $p_r \leftarrow p_g$
      $p_g \leftarrow$ Aggregate(p_t, $t \in s_t$; p_g, $g \in$ G)
      If $t\%h_t == 0$ **do**
        for each participant $k \in s_g$ **do**
          $E_k \leftarrow$ avg(validation results of k in recent R_e rounds)
        $L_g \leftarrow$ Segment(E_k, $k \in s_g$; L_g)
Execute(p_t, D_t)**:**
  $p_t \leftarrow p_g$
  $D_b \leftarrow$ (split D_t into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in D_b$ **do**
      $p_t \leftarrow p_t - \eta \bigtriangledown l(p_t; b)$
  return p_t to server
Aggregate($p_t, t \in s_t; p_g, g \in G$)**:**
  $G_{others} \leftarrow$ (the other global models except for the current one)
  $p_g \leftarrow \alpha \cdot p_g + \beta \cdot \text{avg}(p_t, t \in s_t) + \gamma \cdot \text{avg}(p_o, o \in G_{others})$
  return p_g to server
Segment($E_k, k \in s_g; L_g$)**:**
  $e_k \leftarrow \text{sigmoid}(E_k - \text{avg}(E_k, k \in s_g))$
  if $e_k <$ threshold
    L_g (remove participants k from the current global model)
    initialize p_{new}
    L_g (attach k to the newly initialized global model p_{new})
  return L_g to server

parameters of the other global models with varying component ratios, defined in (7).

$$p_t = \alpha \cdot p_{t-1} + \beta \cdot \frac{\sum_{i=1}^{n} p_i}{n} + \gamma \cdot \frac{\sum_{i=1}^{m} q_i}{m} (\alpha + \beta + \gamma = 1) \quad (7)$$

where p_t represents the updated global parameters, p_{t-1} represents the former global parameters, p_i represents the average-aggregated local parameters, q_i represents the average-aggregated parameters of the other global models, n is the number of participants who conducted local model training in this round, and m is the number of the other global models. α, β and γ represent the ratios of each component respectively, with a sum of 1.

Considering the balance of each component for the update and the stability of Segmented-FL, we employed 0.1 as β, 0.01 as γ, and consequently $(0.9 - 0.01 \cdot m)$ as α. The intact algorithm of Segmented-FL is shown as Algorithm 1.

FIGURE 6. The massively distributed LANs applied in the experiments.

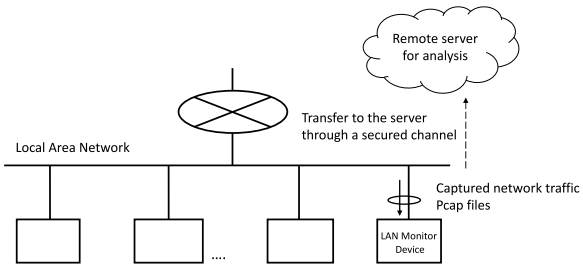


FIGURE 7. The scheme of local network traffic collection: a monitor device was employed for collecting and transporting data in a LAN to the server.

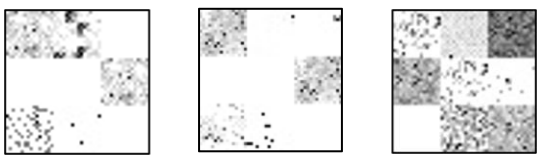


FIGURE 8. Samples of the generated feature maps of network traffic.

IV. EVALUATION

A. EXPERIMENT SETTING

We adopted the experiment data of 20 participants from the LAN-Security Monitoring Project (Fig. 6). In this project, a device connected to a port of the switching hub or the router was used for collecting network traffic in the LAN (Fig. 7). Then the collected data was compressed, encrypted, and transported to the central server daily. The collected traffic data in a LAN includes broadcasts and direct traffic to the device. The Network Time Protocol (NTP) was employed for clock synchronization between a device and the server. We studied and applied a total of 60 days' network traffic data from 20 participants' LANs, from 1st October to 29th November in 2019.

By employing the aforementioned visual analytics, we generated the feature maps from collected network traffic data (Fig. 8).

To evaluate the adaptivity of Segmented-FL in typical applications, we have studied three types of virtual ground truths. This is because we cannot obtain the actual ground truth (i.e., manually labeled data) in the current phase. However, if Segmented-FL could adapt to many virtually generated ground truths, it says that it is possible to adapt

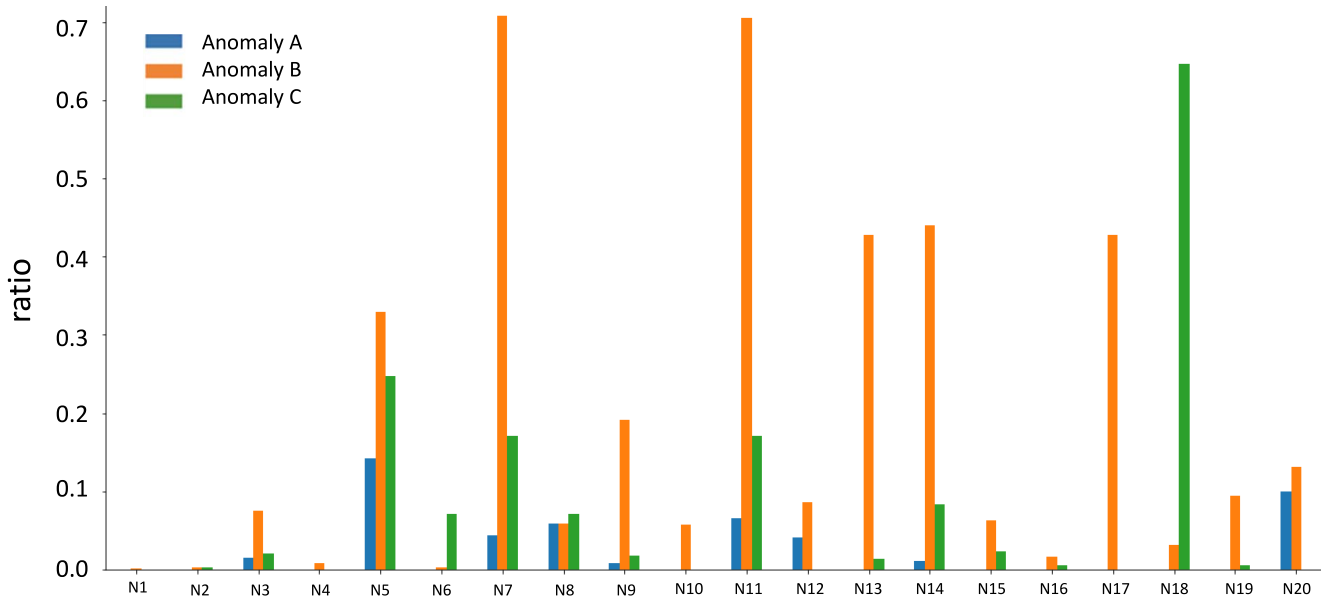


FIGURE 9. The ratios of each type's anomaly to all feature maps generated from network traffic data for each node.

TABLE 1. Knowledge-based labeling.

Methods	Knowledge Type	Definition
A	Server Message Block (SMB) attacks	Detection of any SYN445 to the monitor device.
B	TCP SYN flood attacks	TCP SYN from the same MAC address for more than three times within the time window of 128 seconds.
C	UDP unicast attacks	Detection of any UDP unicast to the monitor device (except NTP with a source port of 123 and DNS with a source port of 53).

to the actual ground truth when we obtain manually labeled data. We admit that a knowledge-based labeling method is a general approach to label possibly malicious data, doesn't certainly fit all participants' data. The employed three types of knowledge-based labeling methods are shown in Table 1.

First, for Method A, SYN445 (TCP port 445) is used to operate Server Message Block (SMB) directly over TCP/IP, where SMB is used for file sharing. However, this port is usually blocked or disabled. The TCP port 445 of a monitor device was opened for luring underlying attacks through SMB. Considering a deployed monitor device in a LAN doesn't possess a purpose of file sharing, detected SYN445 to the monitor device would be malicious.

Second, for Method B of a TCP SYN flood attack, an attacker usually sends repeated SYN packets to every port of the target. The target receives multiple requests to establish communication and responds to each request with an SYN-ACK packet from each open port, unaware of the attack. As the target's connection overflows, requests from other users would be denied, and the target could malfunction. As a result, we extracted MAC addresses of users in a LAN and considered the users sending three or more TCP SYN for requesting a connection with another user in the LAN during the time window of 128 seconds as malicious.

Finally, for Method C, unicast is communication where information is sent from one point to another point directly in a network, and UDP is used to send datagrams. However, due to the monitor device not possessing a purpose of direct datagram sharing with other users, UDP unicast communications to the monitor device would be considered malicious, except for fundamental functions of DNS with a source port of 53 and NTP with a source port of 123.

The ratios of results with each labeling method for 20 participants are shown above (Fig. 9). The graph shows the varying and imbalanced data compositions of participants. The constitution of the generated dataset for Segmented-FL is shown below (Table 2).

Then, we employed Segmented-FL and the four-layer CNN as the basis of our scheme. The scheme conducted collaborative learning on a daily basis, which means for each round, participants conducted local model training based on the same day's data from the datasets. For each round, a specific number of participants conducting local model training were selected on a rolling basis. The reason for the rolling basis is to eliminate the imbalance between users' training times, thus bringing to more precise model evaluation and participant segmentation.

The batch learning method was applied, where the dataset was partitioned into small chunks for training purposes. Usually, a smaller batch size brings out a more satisfactory model at a slower speed, in contrast, a larger one can speed up the training process, albeit bringing out poorer training results.

TABLE 2. Dataset profile.

Participants	Method A		Method B		Method C		Participants	Method A		Method B		Method C	
	Benign	Malicious	Benign	Malicious	Benign	Malicious		Benign	Malicious	Benign	Malicious	Benign	Malicious
N001	46575	0	46497	78	46562	13	**N011**	43531	3044	13763	32812	38600	7975
N002	44512	0	44386	126	44381	131	**N012**	44688	1887	42568	4007	46569	6
N003	47138	706	44230	3614	46847	997	**N013**	46575	0	26644	19931	45950	625
N004	45589	0	45223	366	45585	4	**N014**	46073	502	26081	20494	42662	3913
N005	39140	6502	30639	15003	34364	11278	**N015**	45393	9	42521	2881	44363	1039
N006	46544	31	46433	142	43260	3315	**N016**	45949	3	45190	762	45689	263
N007	44541	2034	13639	32936	38613	7962	**N017**	46575	0	26652	19923	46575	0
N008	42248	2612	42228	2632	41654	3206	**N018**	46047	0	44604	1443	16303	29744
N009	46196	379	37671	8904	45747	828	**N019**	46113	2	41767	4348	45840	275
N010	46575	0	43913	2662	46575	0	**N020**	41925	4650	40489	6086	46567	8

A dataset separating from the datasets for collaborative learning was used to initialize the global model based on the four-layers CNN model with a learning rate of 0.00001, a batch size of 200, and an epoch of five. The dataset for initialization included 782 benign feature maps and 1186 malicious feature maps, which were generated from participant N001's network traffic in September 2019. Besides, for local model training, we applied a learning rate of 0.00001, a batch size of 50, and an epoch of one. We employed the dataset consisting of sixty days' network traffic data to conduct experiments. Since for each round, participants applied data from one day, a total of sixty rounds' adaptive collaborative learning were conducted based on the dataset.

B. VARYING HYPERPARAMETERS OF SEGMENTED-FL

Precision, recall, and f1 score formed the basis of our metrics for evaluating the performance of the scheme, defined as (8). Precision shows the fraction of relevant feature maps successfully classified among all data in the validation set; recall shows the fraction of ones successfully classified among existing relevant feature maps. And the f1 score shows overall performance. Besides, these metrics were weighted by support, namely the number of true instances for each label, to deal with data imbalance.

$$
\begin{aligned}
Weight_P &= \frac{TP+FN}{TP+FP+TN+FN} \\
Weight_N &= \frac{TN+FP}{TP+FP+TN+FN} \\
&Weighted-Precision \\
&= \frac{TP}{TP+FP} \times Weight_P + \frac{TN}{TN+FN} \times Weight_N \\
&Weighted-Recall \\
&= \frac{TP}{TP+FN} \times Weight_P + \frac{TN}{TN+FP} \times Weight_N \\
&Weighted-F1\ Score \\
&= \frac{2 \times Weighted-Precision \times Weighted-Recall}{Weighted-Precision + Weighted-Recall}
\end{aligned} \tag{8}
$$

where TP (True Positives) indicates the number of correct malicious network flows classified by the model in the data, FP (False Positives) indicates the number of incorrect malicious network flows classified in the data, TN (True Negatives) indicates the number of correct benign network flows classified by the model in the data, and FN (False Negatives) indicates the number of incorrect benign network flows classified in the data.

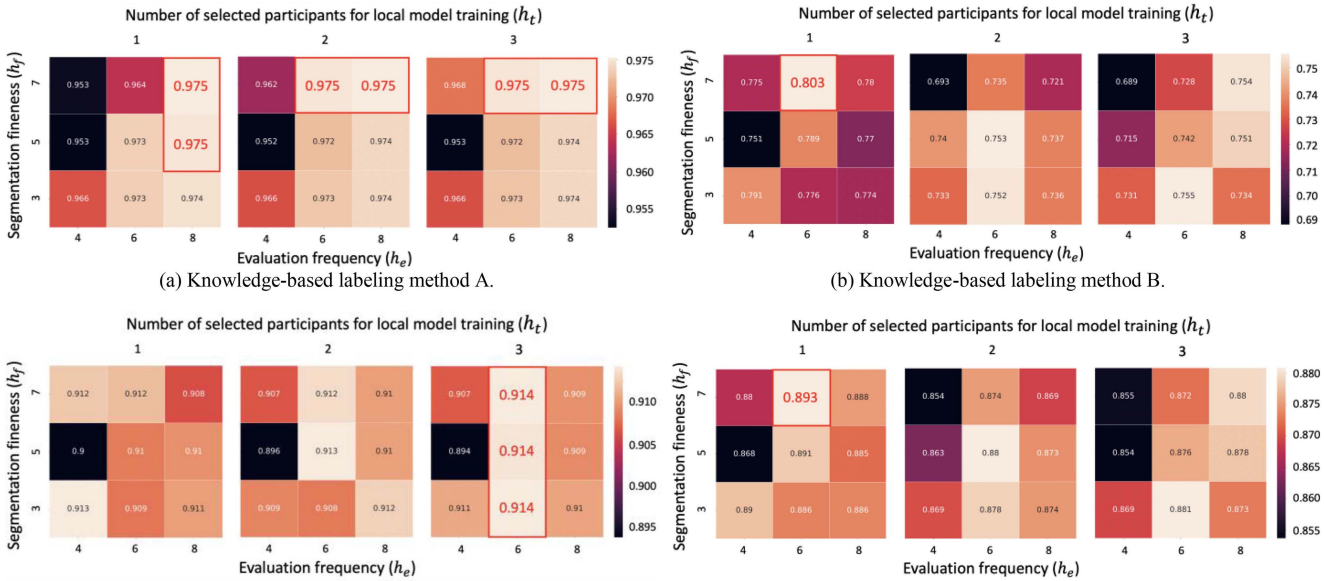
TABLE 3. Varying hyperparameters of segmented-FL.

Hyperparameter*		
Factor of selected participants for local model training (h_t)	Evaluation frequency (h_e)	Segmentation fineness (h_f)
1	4	3
2	6	5
3	8	7

*The hyperparameters are combined to find the best solution.

To optimize the scheme's architecture, we experimented with various hyperparameters to effect performance changes, including the factor of selected participants, evaluation frequency, and segmentation fineness (Table 3). Equation (9) was employed to compute the number of participants for local model training in a group from the factor parameter h_t.

$$N_t = floor\left(\frac{max(N, h_t)}{h_t}\right) \tag{9}$$



(a) Knowledge-based labeling method A.

(b) Knowledge-based labeling method B.

(c) Knowledge-based labeling method C.

(d) Overlapped average results of three methods.

FIGURE 10. Heatmaps of validation weighted F1 scores with various combinations of the hyperparameters.

where N_t is the number of participants for local model training, N is the total number of participants in the group, h_t is the factor parameter, max is a function to compute the maximum between N and h_t, and floor is a function to obtain the greatest integer less than or equal to the input.

We tuned a total of 27 combinations of the various hyperparameters for each labeling method. By comparing the average validation weighted F1 score within the last several rounds (according to the evaluation frequency, e.g., an evaluation frequency of four brings the last four rounds' average), we evaluated the performance of Segmented-FL. The evaluation results for the three types of labeling methods are shown above (Fig. 10). In addition, we applied the F1 score as the metric for the periodic local model evaluation.

We employed the average evaluation results of 20 participants in the last h_e rounds (the last evaluation cycle) as the final performance. Based on the study on the performance of Segmented-FL for various combinations of the hyperparameters, we selected an evaluation frequency of six, a segmentation fineness of seven, and a factor of selected participants for local model training of one. The averaged validation results of 20 participants at each round of Segmented-FL with the selected hyperparameters are shown below (Fig. 11).

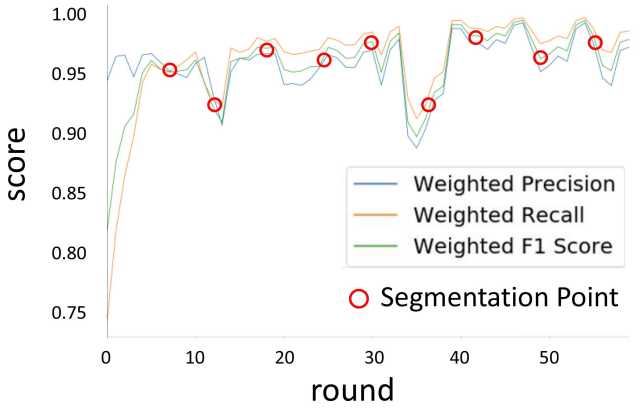
C. COMPARISON WITH FEDERATED LEARNING FOR INTRUSION DETECTION IN MASSIVELY DISTRIBUTED NETWORKS

We employed FL on the dataset with a factor of selected participants for local model training of one, which is the same as the selected hyperparameter of Segmented-FL. Then, the metrics of weighted precision, recall, and F1 score were used to evaluate the performance with the three knowledge-based labeling methods. The average of the last six rounds' evaluation results was used as the final performance of FL. Besides, considering the situation that for each round not all of the participants would conduct local model training ($h_t \neq 1$), we also studied the performance of FL and Segmented-FL when the factor of selected participants was two. A comparison was conducted between the performance of FL and the proposed method, with the two different settings of hyperparameters (Fig. 12).

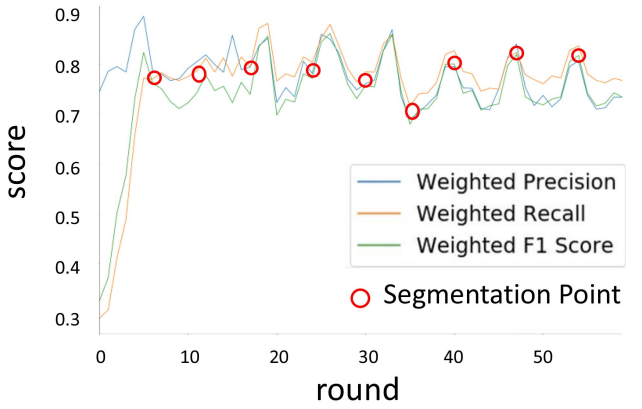
As shown in the graphs, for each type of method, the average weighted F1 score based on validation sets had an increase. For Method B which showed relatively disadvantaged performance, the proposed method with the optimized hyperparameters contributed an increase up to 4.0% in the average weighted F1 score based on validation sets. When the factor parameter h_t was two, the gain rose to 4.8%. That's because a larger h_t brought to fewer participants for local model training at each round, contributing to a considerable decrease in the performance of FL. For Method A and Method C, though the obtained gains were not so large, the increases in performance were still sufficient since they showed relatively high validation scores. Secondly, data constitution imbalance in the networks let the weighted metrics put more emphasis on the validation results from benign data with these two methods. The comparison result shows Segmented-FL has better performance for intrusion detection in large-scale massively distributed networks.

V. DISCUSSION

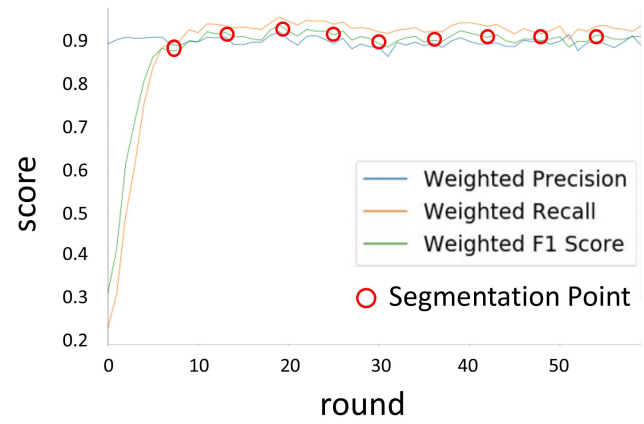
Segmented-FL has a feature of automatic architecture transformation based on the periodic evaluation, thus adapting to diverse data from massively distributed networks. Compared with FL, it showed more adaptability to collaborative learning with imbalanced and diverse data. The labeling methods



(a) Knowledge-based labeling method A.

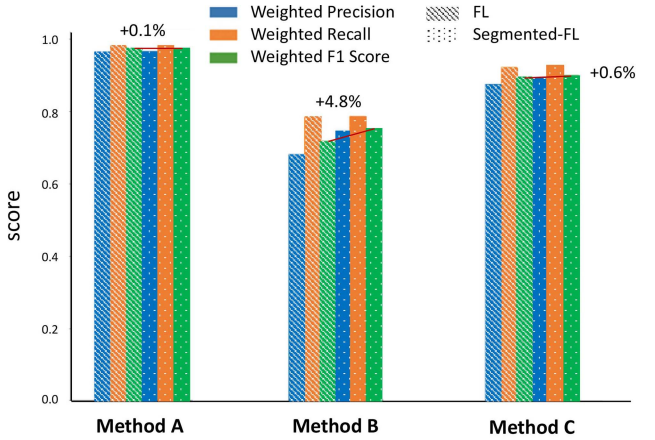


(b) Knowledge-based labeling method B.

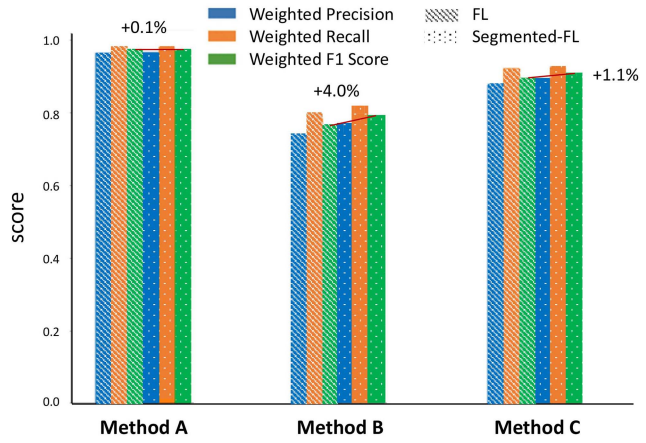


(c) Knowledge-based labeling method C.

FIGURE 11. The averaged validation results of 20 participants at each round of Segmented-FL with the optimized hyperparameters.



(a) When the factor of selected participants for local model training is one.



(b) When the factor of selected participants for local model training is two.

FIGURE 12. A comparison between the performance of FL and Segmented-FL against a range of metrics including weighted precision, recall, and F1 score, when applying the two different hyperparameter settings.

employed were based on network traffic patterns within a small time slot of 128 seconds, which allowed us to detect malicious behavior precisely. Besides, the visualization of network traffic by feature maps provided more explainability for the detection of malicious behavior.

Several considerations of future improvement of the scheme include: a feature map was generated based on frequency information of several network protocols, however, other information such as packet length and payload size could also include hidden features of malicious behaviors; the experiment traffic data was observed by a normal host connected to a LAN (broadcasts and direct packets), hence, a combination with upstream captured network traffic would provide more insights into underlying malicious traffic features; α, β, and γ were employed to represent the ratio of each component for aggregation, further discussion on these parameters could be considered.

VI. CONCLUSION

We proposed Segmented-FL to solve the problem of a network's traffic data not always fitting into the single global model of FL situation. This research is focusing on the study of how Segmented-FL works for intrusion detection in real-world massively distributed LANs. By studying the optimized hyperparameters of Segmented-FL and employing three evaluation methods, the validation result shows that Segmented-FL has better performance in all three types

of intrusion detection tasks, achieving validation weighted F1 scores of 0.964, 0.803, and 0.912 with Method A, Method B, and Method C respectively. For each method, this scheme shows a gain of 0.1%, 4.0% and 1.1% in performance compared with FL.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, and B. A. Arcas, "Federated learning of deep networks using model averaging," 2016. [Online]. Available: arXiv:1602.05629v1.

[2] A. Abeshu and N. Chilamkurti, "Deep learning: The Frontier for distributed attack detection in fog-to-things computing," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 169–175, Feb. 2018.

[3] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019, doi: 10.1109/JSAC.2019.2904348.

[4] *LAN-Security Monitoring Project*. Accessed: Oct. 5, 2020. [Online]. Available: http://www.lan-security.net

[5] U. K. Premaratne, J. Samarabandu, T. S. Sidhu, R. Beresh, and J.-C. Tan, "An intrusion detection system for IEC61850 automated substations," *IEEE Trans. Power Del.*, vol. 25, no. 4, pp. 2376–2383, Oct. 2010.

[6] M. D. Santo, A. Vaccaro, D. Villacci, and E. Zimeo, "A distributed architecture for online power systems security analysis," *IEEE Trans. Ind. Electron.*, vol. 51, no. 6, pp. 1238–1248, Dec. 2004.

[7] A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. N. Fovino, and A. Trombetta, "A multidimensional critical state analysis for detecting intrusions in SCADA systems," *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 179–186, May 2011.

[8] S. Shin, T. Kwon, G.-Y. Jo, Y. Park, and H. Rhy, "An experimental study of hierarchical intrusion detection for wireless industrial sensor networks," *IEEE Trans. Ind. Informat.*, vol. 6, no. 4, pp. 744–757, Nov. 2010.

[9] C.-H. Tsang and S. Kwong, "Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction," in *Proc. Int. Conf. Ind. Technol.*, Hong Kong, 2005, pp. 51–56.

[10] N. Mishra and S. Mishra, "Support vector machine used in network intrusion detection," in *Proc. Nat. Workshop Internet Things (IoT)*, 2018, pp. 25–27.

[11] T. Omrani, A. Dallali, B. C. Rhaimi, and J. Fattahi, "Fusion of ANN and SVM classifiers for network attack detection," in *Proc. 18th Int. Conf. Sci. Techn. Autom. Control Comput. Eng.*, 2017, pp. 12–25.

[12] M. A. Salama, H. F. Eid, R. A. Ramadan, A. Darwish, and A. E. Hassanien, "Hybrid intelligent intrusion detection scheme," in *Soft Computing in Industrial Applications*. Berlin, Germany: Springer, 2011, pp. 293–303.

[13] J. Yang, J. Deng, S. Li, and Y. Hao, "Improved traffic detection with support vector machine based on restricted Boltzmann machine," *Soft Comput.*, vol. 21, no. 11, pp. 3101–3112, 2017.

[14] P. H. Duy and N. N. Diep, "Intrusion detection using deep neural network," *Southeast Asian J. Sci.*, vol. 5, no. 2, pp. 111–125, 2017.

[15] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, 2015, pp. 11–20.

[16] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 3854–3861.

[17] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning, session C3: Machine learning privacy," in *Proc. CCS*, 2017, pp. 603–618.

[18] J. Schneible and A. Lu, "Anomaly detection on the edge," in *Proc. MILCOM IEEE Mil. Commun. Conf. (MILCOM)*, Baltimore, MD, USA, 2017, pp. 678–682.

[19] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," in *Proc. 10th Int. Symp. Inf. Commun. Technol. (SoICT)*, New York, NY, USA, 2019, pp. 273–279. [Online]. Available: https://doi.org/10.1145/3368926.3369705

[20] H. Daga, P. K. Nicholson, A. Gavrilovska, and D. Lugones, "Cartel: A system for collaborative transfer learning at the edge," in *Proc. ACM Symp. Cloud Comput. (SoCC)*, New York, NY, USA, 2019, pp. 25–37.

[21] 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. Security Privacy*, 2018, pp. 108–116.

[22] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Trans. Intell. Transp. Syst.*, early access, Aug. 31, 2020, doi: 10.1109/TITS.2020.3017474.

[23] Y. Sun, H. Ochiai, and H. Esaki, "Visual analytics for anomaly classification in LAN based on deep convolutional neural network," in *Proc. IEEE Int. Conf. Informat. Electron. Vis.*, 2020, pp. 7–12.

[24] Y. Sun, H. Ochiai, and H. Esaki, "Detection and classification of network events in LAN using CNN," in *Proc. IEEE Int. Conf. Inf. Technol.*, 2019, pp. 203–207.

[25] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security—A survey," 2017. [Online]. Available: arXiv:1701.04525v1.

[26] L. Goodfellow, Y. Bengio, and A. Courvile, *Deep Learning (Adaptive Computation and Machine Learning)*. Cambridge, MA, USA: MIT Press, 2016

[27] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[28] A. Elsaeidy, K. S. Munasinghe, D. Sharma, and A. Jamalipour, "Intrusion detection in smart cities using restricted Boltzmann machines," *J. Netw. Comput. Appl.*, vol. 135, pp. 76–83, Jun. 2019.

YUWEI SUN received the B.E. degree in computer science and technology from North China Electric Power University in 2018. He is currently pursuing the Ph.D. degree with the Graduate School of Information Science and Technology, University of Tokyo. In 2020, he was the Fellow of the ASP, Massachusetts Institute of Technology. Since 2019, he has been working with the Campus Computing Centre, United Nations University Centre on Intelligent Networking Systems.

HIROSHI ESAKI (Member, IEEE) received the Ph.D. degree from the University of Tokyo, Japan, in 1998. In 1987, he joined Research and Development Center, Toshiba Corporation. From 1990 to 1991, he was with Applied Research Laboratory, Bell-Core, Inc., Murray Hill, NJ, USA, as a Residential Researcher. From 1994 to 1996, he was with the Center for Telecommunication Research, Columbia University, New York, NY, USA. Since 1998, he has been serving as a Professor with the University of Tokyo, and as a Board Member of WIDE Project. He is currently the Executive Director of IPv6 Promotion Council, the Vice President of JPNIC, IPv6 Forum Fellow, and the Director of WIDE Project.

HIDEYA OCHIAI (Member, IEEE) received the B.E., M.E., and Ph.D. degrees from the University of Tokyo, Japan, in 2006, 2008, 2011, respectively, where he is an Associate Professor. He is involved in the standardization of facility information access protocol in IEEE 1888, ISO/IEC, and ASHRAE. His research interests have been sensor networking, delay tolerant networking, and building automation systems, IoT protocols, and cyber-security.