Intrusion Detection with Segmented Federated Learning for Large-Scale Multiple LANs

Yuwei Sun
Graduate School of Information Science and Technology The University of Tokyo
Tokyo, Japan
sywtokyo@hongo.wide.ad.jp

Hideya Ochiai
Graduate School of Information Science and Technology The University of Tokyo
Tokyo, Japan
jo2lxq@hongo.wide.ad.jp

Hiroshi Esaki
Graduate School of Information Science and Technology The University of Tokyo
Tokyo, Japan
hiroshi@wide.ad.jp

Abstract—**Traditional approaches to cybersecurity issues usually protect users from attacks after the occurrence of specific types of attacks. Besides, patterns of recent cyberattacks tend to be changeable, which add up to unpredictability of them. On the other hand, machine learning, as a new method used to detect intrusion, is attracting more and more attention. Moreover, through the sharing of local training data, the centralized learning approach has proven to improve a model's performance. In this research, a segmented federated learning is proposed, different from a collaborative learning based on single global model in a traditional federated learning model, it keeps multiple global models which allow each segment of participants to conduct collaborative learning separately and rearranges the segmentation of participants dynamically as well. Furthermore, these multiple global models interact with each other for updating parameters, thus being adaptable to various participants' LANs. A dataset covering two months' traffic data from 20 participants' LANs in the LAN-Security Monitoring Project is used. We adopt three types of knowledge-based methods for labeling network events and train a CNN model based on the dataset. At last, we achieve validation accuracies of 0.923, 0.813 and 0.877 individually with these labeling methods.**

Keywords—LAN, cybersecurity, machine learning, segmented federated learning, CNN

I. INTRODUCTION

In former research of the LAN-Security Monitoring Project [1], a smart device is designed and connected to a router in a LAN, thus collecting and transporting network traffic data securely to the central server. Moreover, the collected traffic data include broadcast data in a LAN and any communication directly sent to the device. Then an algorithm at the central server is used to detect abnormal behavior inside LANs based on these data. In total, more than fifty devices have been set up mainly in LANs of university laboratories and research institutes. In addition, all data are packaged and transported to the central server on a daily basis, which means each data file includes one day's traffic data in a LAN.

Considering the privacy of participants transporting their local data and diversity of their LANs, a segmented federated learning is proposed in this research, which allows participants to share parameters of local training models instead of original data as well as adjusting itself according to variance among LANs. Through parameters sharing and structure transformation, this scheme is supposed to improve the overall performance of intrusion detection in large-scale networked LANs. This scheme consists of two main parts: intrusion detection in LANs and segmented federated learning.

First, to detect malware in the network, a discriminator consisting of nine types of protocol information is used in this research, as follows: ARP, IP, TCP, UDP, HTTP, HTTPS, mDNS, DHCP, and Others. After that, we use a structure called the Hilbert curve to convert these various types of information into a feature map based on the frequency of each type's communication, representing a network event defined in 128 seconds. Then we adopt three types of knowledge-based approaches for labeling these network events, which include detection of a SYN445 to the monitor device, detection of TCP SYN from an IP with a frequency of more than three times, and detection of a UDP unicast to the monitor device (except the case of NTP communications between the monitoring device and the central server and the case of DNS communications).

Then, a four-layer convolutional neural network model, consisting of two convolution layers, each of which is followed by a maxpooling layer, and two fully-connected layers, is adopted and trained based on the local dataset, with feature maps as the input data and types of network events obtained by knowledge-based approaches as the labels. Furthermore, we use a learning function called RMSProp and mini-batch learning in this model. After training, we are supposed to obtain a model that can tell if a network event is malicious or not through the classification of its feature map.

Moreover, a segmented federated learning method is adopted in this research for parameters sharing among participants as well as adapting itself to various networks. Here, for every day, the model conducts a learning progress, called a round. For each round, selected participants will train local models using local datasets, which are generated based on a specific day's traffic data in participants' networks. Furthermore, for every six rounds, the system will conduct a performance evaluation of each participant under the current global branch using the variance of all participants' average accuracies during the recent six rounds, which is used to decide if a participant continues staying at the current branch. On the other hand, if it shows a result under a specific threshold, this one will be transferred to another branch, which is initialized accompanied by the judgment. Consequently, from the next round, these participants will train on their dataset under different global models.

Furthermore, we use a dataset that is separated from the dataset for the segmented federated learning to initialize a global model, with a learning rate of 0.00001, a batch size of 200, and an epoch of five. This dataset includes 782 benign feature maps and 1186 malicious feature maps. On the other hand, we train local models at each round with a learning rate of 0.00001, a batch size of 50, and an epoch of one. Then, for each round, parameters of selected participants' models will be uploaded to the central server. Then, the corresponding global model conducts aggregation based on these parameters, including ones of a former global model, and ones of the other global models, each of which is with different ratios, to update the global model. After that, all participants under this global model will download the updated global model to replace their local models.

For evaluation of this monitor system, we record accuracy of all participants for intrusion detection with each knowledge-based labeling method and visualize the segmentation of the system when it encounters with participants with relatively disadvantaged performance under a global model.

This paper is organized as follows. Section 2 discusses related works about intrusion detection with approaches of machine learning and applications of the federated learning on cybersecurity issues. Section 3 provides an overview of the scheme, including representation of network events, classification with the CNN, and parameters sharing with the segmented federated learning. Section 4 presents the performance evaluation of this system based on the precision, visualizing global models' segmentation with various knowledge-based labeling methods. Section 5, we conclude the paper and give out the future work of this research.

II. Related Work

Intrusion detection has been a new trend in the cybersecurity area nowadays. Especially, in a local area network (LAN), the number of monitored intrusion is increasing. Traditional approaches to this issue include an application of several knowledge-based rules on network communication, and once these rules are satisfied, a network event will be considered as malicious. Moreover, several traditional machine learning methods such as support vector machine (SVM) and neural network (NN) have been used to address issues of network attacks detection in personal computers and critical infrastructure [2][3]. However, due to the limitations of these methods on dealing with big data and adaptability to various network environments, they have shown disadvantages in solving complicated detection problems.

On the other hand, since the advancement of deep learning in recent years, a large-scale data analysis on network traffic data has become possible and shown great performance as well. For instance, Salama et al. [4] presented an intrusion detection hybrid scheme using deep belief network (DBN) and SVM, classifying the intrusion into two clusters: normal or attack. They adopt DBN for reducing the dimension of features and SVM for the classifier. They evaluated their scheme with the NSL-KDD dataset [5] and achieved an accuracy of above 0.9 at last. Moreover, in another research conducted by Yang et al. [6], they use restricted Boltzmann machine (RBM) to extract high-level feature representation of traffic data and train SVM with stochastic gradient descent (SGD) for classification of it. Duy et al. [7] discriminated the application of feedforward neural network (FFNN) on network intrusion detection based on the NSL-KDD dataset. Their model achieves an F_1 score of 0.962 for evaluation. In addition, this FFNN model includes four hidden layers of 60 neurons, using an activation function of ReLU and a learning rate of 0.001 for training. Furthermore, Saxe et al. [8] proposed a deep neural network (DNN) based malware detector that employs two-dimensional binary program features to detect malware. Yousefi-Azar et al. [9] gave out a generative feature learning-based approach for malware classification, where latent features from the hidden layer of autoencoder (AE) are adopted for anomaly detection.

Moreover, as former research, a centralized learning, where local training data of participants are transported to a central server for centralized learning, is used to improve the performance of intrusion detection with machine learning [10]. However, several privacy issues accompanying the transportation progress have been given up. On the contrast, the federated learning (FL) is proposed to access these issues, by allowing participants to achieve the purpose of collaborative learning without sharing their private local data, instead, sharing local model parameters. Furthermore, there have been several pieces of research focusing on the application of the federated learning on cybersecurity issues. For example, Abeshu et al. [11] proposed a cyberattack detection model using FL, with edge nodes as participants. In this model, to improve the accuracy in detecting attacks, each participant sends its trained model based on a local dataset to the server for parameters sharing. Through this approach, they are aimed to enhance the privacy of participants and reduce the traffic load of networks while transporting as well. Nguyen et al. [12] presented an approach of adopting Internet of things (IoT) gateways as participants and an IoT security service provider as the server node for the aggregation of machine learning model parameters. At last, they achieved an accuracy of 0.956 for intrusion detection in a real-world smart home deployment setting.

However, research mentioned above has limits in intrusion detection in various network environments as well as a stable and resilient learning progress as well. Different from former research, we propose a segmented federated learning, in which the system structure is adjusted automatically according to performance of each participant, thus having great adaptivity to various networks. At the same time, different from adopting traditional machine learning approaches, we use a convolutional neural network, which is one kind of deep learning, for purposes of the extraction of high-level feature representation and network events classification. As a result, this system is supposed to have the ability of dealing with big data of network traffic as well as a collaborative learning with a stable and robust progress.

III. Intrusion Detection in LANs with Segmented Federated Learning

In this research, we extend the traditional federated learning method and apply it on intrusion detection in LAN based on a dataset from 20 participants in the LAN-security Monitoring Project [1]. Instead of uploading local traffic data to a central server, we allow participants to initialize and train a machine learning model in the local. Moreover, through uploading parameters of these trained models, the central server conducts an aggregating function to generate a global model for the intelligence sharing among all participants. For every round, only selected participants retrain their local models, and for every six rounds, the system conducts a performance evaluation of each participant based on the recent six records of validation accuracy of local models. The ones showing relatively disadvantaged performance are removed from the current global model and transferred to a new initialized global model (Fig. 1).

Considering diversity between networks of participants, independently trained local models are greatly different from each other. Through aggregating parameters of these local models as well as other global models, it is aimed to generate models with resilience and adaptivity to intrusion detection issues in various network environments. Furthermore, since original traffic data are kept in the local, the application of a federated-learning-based approach also proves to enhance the privacy of participants in large-scale networked systems.

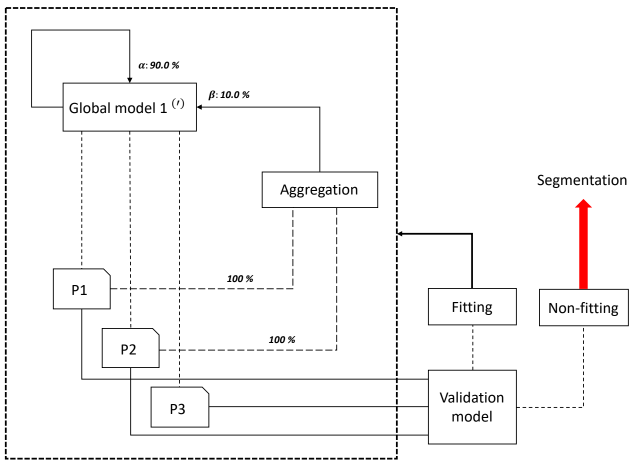


Fig. 1. The initial structure of the large-scale networked monitor systems. P1, P2, and P3 represent participants. A validation model is adopted for every six rounds to evaluate performance of each participant. The participants who show relatively good performance are kept under a current global model, while participants who show relatively poor performance under the current one are removed from the current global model and further processed for segmentation.

A. *Experiment data*

In this research, we adopt experiment data from 20 participants' LANs in the LAN-security Monitoring Project. In this project, a smart device (a node) is connected to a router in a LAN for collecting network traffic data. Then these data are packaged and transported to a central server for anomaly detection on a daily basis. The collected network traffic data include all broadcast traffic in a LAN and any communication sent directly to the devices. Moreover, all data are collected in the format of pcap capture files, and the NTP (Network Time Protocol) is used for clock synchronization between a node and the central server in this project.

For the experiment data in this research, we extract two months' collected network traffic data from 20 participants in the LAN-security Monitoring Project, from 1st October to 29th November 2019, a total of 60 days. Consequently, for these participants, the same amount of traffic data during the same period are adopted in this research.

B. *Feature representation*

To represent features of network traffic data, we adopt communication frequencies of nine types of protocols as discriminators, including the IP, ARP, TCP, HTTP, HTTPS, UDP, mDNS, DHCP and other. In detail, we compute each record using how many packets of each protocol are transmitted or received within a period of 0.5 second. Furthermore, a total amount of 256 records are used to generate one feature map of a specific protocol, and a parameter called fineness is adopted to adjust how finely we consider information hidden in network traffic data. The time period represented by each feature map is expressed as in (1). Through utilizing these parameters (fineness and size), we bring features of traffic data different in recording period and fineness into images with the same size. In this research, we put communication frequency information of protocols into one feature map, with a period of 128 seconds.

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

Where T is the period represented by a feature map, Tst (time standard) is a standard interval for each recording with a value of one second here and s is the size of the generated feature map of each protocol. In this research, the fineness has a value of 0.5 and the s has a value of 16.

Moreover, we further convert these frequency information into pixel values using (2). Then we adopt a structure called the Hilbert curve to project these 256 records in the format of pixel values into specific positions in a feature map with a width and height of 16 pixels, considering the adaptivity of generated feature maps to a machine learning model as well. Here, the Hilbert curve is a method used to transform the structure of data so that it fills up all space in an image.

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

Where p_i represents the corresponding pixel value, c_i shows the communication frequency of each protocol, and c represents all records of the frequency during the period of 128 seconds in one feature map. As a result, $\frac{c_i}{Max(c)}$ has a value in an interval of (0, 1], and p_i has a value in an interval of (0, 255].

Furthermore, we project the feature maps of these nine types of protocols mentioned above into different arears of an upper image through the array exchange, as well as adding an arear for records of the other protocols, to represent features of network traffic data in LAN with one image (Fig. 2). Consequently, it is considered that features of network traffic data in LAN can be represented by a series of time-related feature maps, each of which has a size of 48 × 48. And the generated feature maps using the dataset from 20 participants' networks are shown below (Fig. 3).

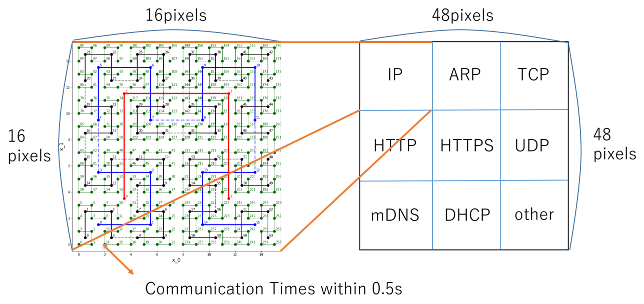


Fig. 2. Feature representation of network traffic data in a LAN based on the Hilbert curve structure: For each protocol, time-related frequency information is converted to pixel value of specific positions; feature maps of nine types of protocol information are projected into specific areas in a feature map for representation of a network event.

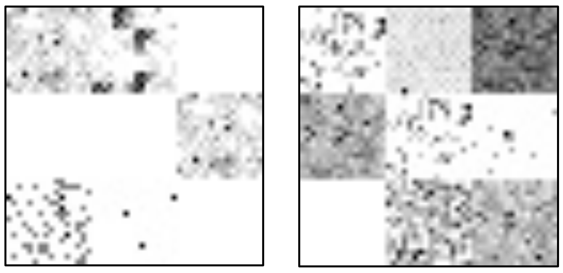


Fig. 3. Samples of network events' feature maps with a size of 48 × 48 (Left: A feature map of N009; right: A feature map of N011).

TABLE Ⅰ. Constitution of the dataset.

Participants	Knowledge A		Knowledge B		Knowledge C		Participants	Knowledge A		Knowledge B		Knowledge 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

C. *Knowledge-based labeling*

In this research, we adopt knowledge-based approaches to label these feature maps of network events, thus dividing them into two clusters, benign or malicious. We consider and select three types of patterns for intrusion detection based on former monitoring and analyzing of participants' network traffic data in the LAN-security Monitoring Project. And the patterns we adopt for labeling are as the following:

- Knowledge A: Detection of any SYN445 to the monitor device
- Knowledge B: TCP SYN from the same IP with a frequency of more than three times
- Knowledge C: Detection of any UDP unicast to the monitor device (except the communications of NTP with a source port of 123 and DNS with a source port of 53)

Then we utilize these labels based on various patterns and the generated feature maps of network events as local training datasets. Moreover, the constitutions of each node's datasets based on three types of labeling patterns are shown above (TABLE Ⅰ)

D. *Classification with Convolutional Neural Network*

We adopt a four-layer CNN model, consisting of two convolution layers, each of which is followed by a maxpooling layer, and two fully-connected layers, to train local datasets mentioned above, where feature maps are used as the input and network event labels are used as the output (Fig. 4). In addition, a padding with a value of one is used in this CNN model. For the first convolution layer, ten kernels with a size of 3 × 3 are adopted with a stride of one. For the second convolution layer, ten kernels with a size of 1 × 1 are adopted with a stride of one. And the fully-connected layer consists of 200 neurons. An output layer with two neurons for classification between benign and malicious is adopted in this model.

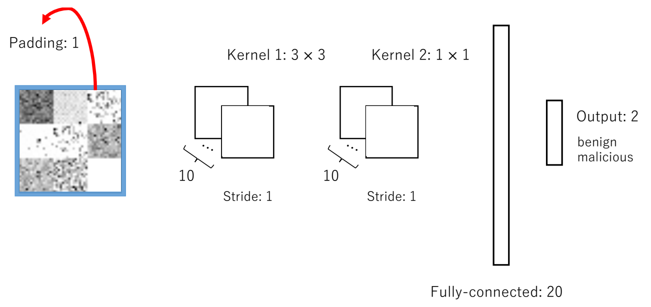


Fig. 4. The structure of the four-layer CNN model, with feature maps as the input and results of knowledge-based labeling as labels.

Moreover, we adopt a learning function called RMSProp in this model, which is defined in the following (3) (4). It 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. Furthermore, softmax is used as the activation function at the last layer to obtain possibilities of each cluster, thus classifying the input of feature maps (5).

$$h_t = \rho * h_{t-1} + (1-\rho) * \frac{\partial L}{\partial W_t} \odot \frac{\partial L}{\partial W_t} \tag{3}$$

$$W_{t+1} = W_t - \eta \frac{1}{\sqrt{h_t + \epsilon}} \odot \frac{\partial L}{\partial W_t} \tag{4}$$

Where L is the loss, W is the weight of the node, and ρ is the decay rate with a value of 0.9.

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{5}$$

Where a standard exponential function is applied to each element x_i of the input vector x, and thus normalizing these values by dividing by the sum of all exponentials. Then, each component including negative, greater than one, or might not sum to 1, will be in the interval (0, 1), with a sum of 1.

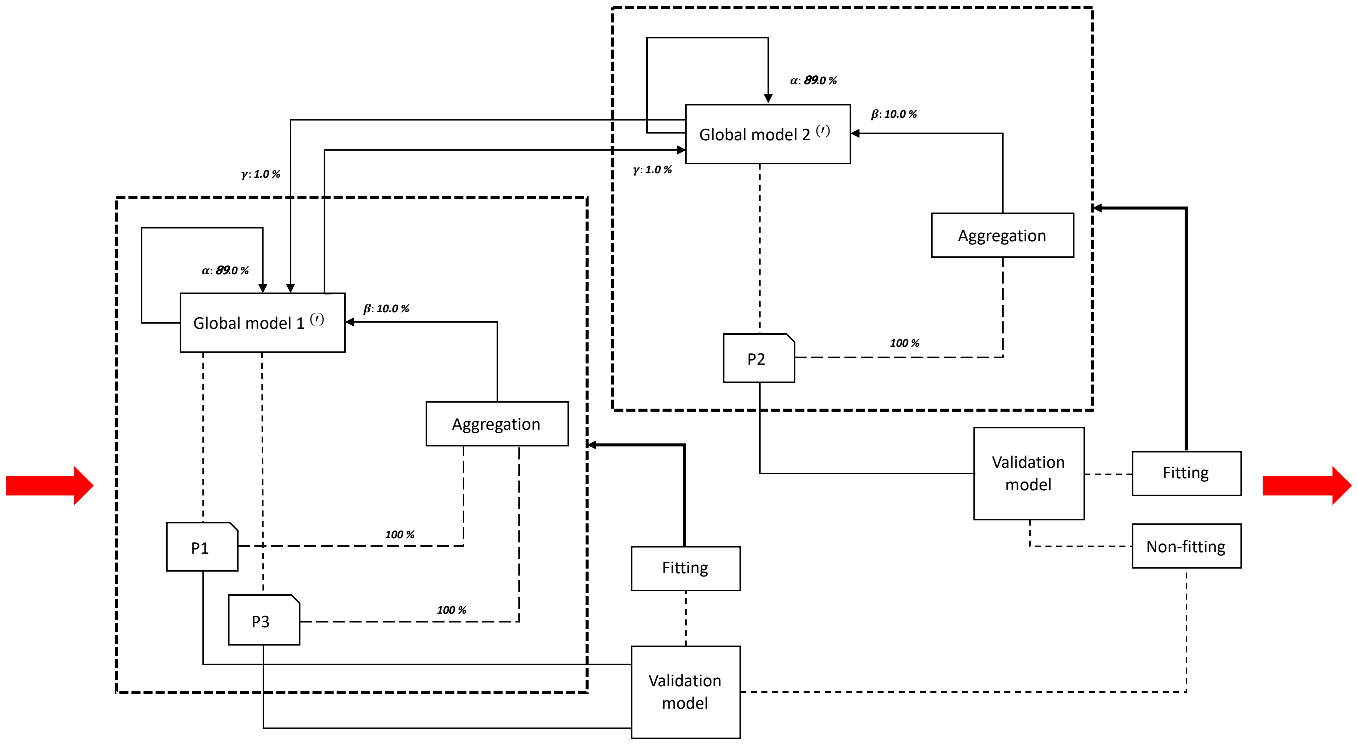


Fig. 6. The scheme of the segmented federated learning when a participant's local dataset doesn't fit the current global model: A new global model is initialized to fit this participant's local dataset. At the same time, parameters of the aggregating algorithm for generating global models of the next round are adjusted automatically based on a new structure of the segmented federated learning model.

E. Segmented Federated Learning

A segmented federated learning model consists of two parts, the central server and participants. Different from a centralized learning model which needs participants to upload local data, in a scheme of the segmented federated learning, participants conduct training locally and share parameters of trained local models with the central server to achieve the purpose of collaborative learning. On the other hand, an aggregator in the central server is adopted to obtain a global model based on the uploaded parameters as well as the former global model. Moreover, the obtained global model is used for updating the participants' local models.

In detail, first, the center server initializes a global model with pre-trained initialization parameters. Then, all participants download the initialized global model to their local networks. After that, for each round, all participants download a global model at the central server to replace their local ones. Then selected participants, using a roll based approach in Algorithm 1, perform retraining of local models using the current day's dataset. After all selected participants complete the retraining, these updated parameters of local models are uploaded to the central server for aggregation. Here, a round is defined as a processing section with a period of one day, from all participants updating their local models to the central server completing the aggregation. As such, at the next round, all participants replace their local models with the updated global models for a new processing section.

Furthermore, since for every two rounds, all participants complete a round of retraining, we conduct performance evaluation for every six rounds (three rounds of retraining) in order to achieve a more precise result. And a validation model defined as (6) and (7), is used to evaluate performance of all participants under the same global model in recent six rounds. The average accuracies of participants during the last six rounds are computed. Then we compute the difference between these average accuracies of participants to evaluate participants' performance. At last, we adopt the Sigmoid function, thus converting the evaluation results into an interval of (0, 1).

$$d_i = acc_i - \frac{\sum_{i=1}^{n} acc_i}{n} \tag{6}$$

$$e_i = \frac{1}{1+e^{-d_i}} \tag{7}$$

Where n is the number of participants, acc_i represents an average test accuracy of participant i for last six rounds, d_i shows the extent of difference between each participant's accuracy and the average of them, and e_i is the output of the Sigmoid function.

Through this approach, the performance of a participant under the current global model is represented with a value in an interval of (0, 1). Then, according to the evaluation result, we remove these participants with a result below the threshold, which is adjusted to a value of 0.47. A new global model is initialized with parameters from the average of these participants' local ones. Then these participants are moved to this newly initialized global model for the next round's processing (Fig. 6). In addition, the threshold here is adjusted and refined, considering that a relatively small threshold leads to a few participants having the possibility to be transplanted to a new global model, and a relatively large one leads to over segmentation of participants.

Moreover, for the aggregation in the central server, we adopt (8) to compute the parameters of global models, based on parameters of various sources including the former global model, uploaded local models of selected participants and the other global models (Fig. 6).

$$p_t = \alpha \cdot p_{t-1} + \beta \cdot \frac{\sum_{i=1}^{n} p_i}{n} + \gamma \cdot \frac{\sum_{i=1}^{m} p_i}{m} \; (\alpha + \beta + \gamma = 1) \quad (8)$$

Where p_t represents new parameters of a global model, p_{t-1} represents parameters of the former global model, p_i shows the uploaded parameters of the participants who conduct training for the current round, q_i represents parameters of the other global models, n is the number of participants conducting local training, and m is the number of the other global models. α, β and γ represent ratios of each component for aggregation, with a sum of 1. Here, considering the balance of parameters updating with respect to various components, β is 0.1, γ is 0.01, and as a result, α has a value of (0.9 - 0.01 $\cdot$ m). And the intact algorithm for the segmented federated learning is defined as Algorithm 1.

Algorithm 1 *SEGMENTED FEDERATED LEARNING.* N_p is the total number of participants. D_t is the local dataset of participant *t*. *G* is all global models. L_g is a list including segment information of participants. *B* is the batch size. *E* is the epoch. α, β and γ are ratios of each component for aggregation. *threshold* is influenced by fineness for segmentation.

```
Server executes:
  initialize $p_0$
  for each round $t$ = 1, 2, . . . do
    for each global model g = 1, 2, . . . do
      $N_t \leftarrow (N_g+1)$ // 2
      $B_t \leftarrow$ (split $N_g$ into batches of size $N_t$)
      $s_t \leftarrow B_t[t$ % 2]
      $s_r \leftarrow$ (set of participants without local 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 $t$ % 6 == 0 do
        for each participant $k \in s_g$ do
          $acc_k \leftarrow$ mean(validation accuracy of $k$ in recent six rounds)
        $L_g \leftarrow$ Segment($acc_k$, $k \in s_g$; $L_g$)

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
      $p_t \leftarrow p_t - \eta \nabla l(p_t; b)$
  return $p_t$ to server

Aggregate($p_t$, $t \in s_t$; $p_g$, $g \in$ G):
  $G_o \leftarrow$ (the other global models except the current one)
  $p_g \leftarrow \alpha \cdot p_g + \beta \cdot mean(p_t, t \in s_t) + \gamma \cdot mean(p_o, o \in G_o)$ $(\alpha + \beta + \gamma = 1)$
  return $p_g$ to server

Segment($acc_k$, $k \in s_g$; $L_g$):
  $e_k \leftarrow$ sigmoid($acc_k$ − mean($acc_k$, $k \in s_g$))
  if $e_k$ < threshold
    $L_g \leftarrow$ (remove all participants $k$ from the current global model)
    $L_g \leftarrow$ (remove this global model if there are no participants left)
    initialize $p_{new}$
    $L_g \leftarrow$ (attach $k$ to a newly initialized global model)
  return $L_g$ to server
```

IV. Evaluation

In this research, we adopt a batch size of 200 and an epoch of one to train local models. We adopt a dataset consisting of sixty days' traffic data collected from 20 participants in the LAN-security Monitoring Project to evaluate our scheme. Moreover, as discussed above, a round is conducted with a period of one day, hence a total of sixty rounds are conducted for the segmented federated learning in this research.

For performance evaluation of the segmented federated learning, we adopt the presicion defined as (9) to show the adpativity of each local model to the correspomding global model. Moreover, the results of validation are all computed at the beginning of each round. The corresponding test accuracy of 20 participants when adopting three types of knowledge-based labeling methods are shown below (Fig. 7, Fig. 8, Fig. 9). In addition, as discussed above, these results are also used for the judgement of segmentation for every six rounds.

$$Precision = \frac{TP}{TP+FP} \quad (9)$$

Where TP (True Positives) indicates the number of feature maps of network events successfully classified by a local CNN model, and FN (False Negative) represents the number of unsuccessfully classified ones.

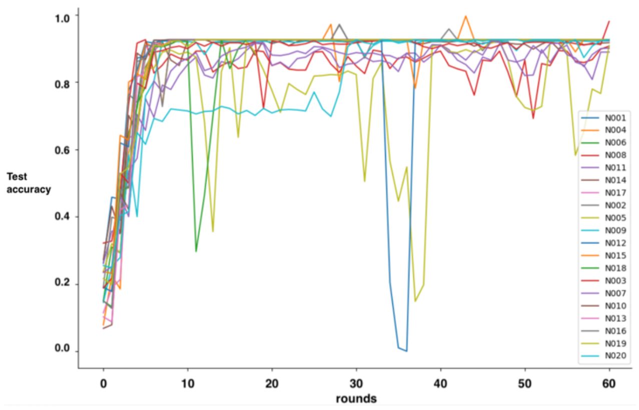


Fig. 7. Test accuracy with Knowledge A: Almost all participants' local models achieve a stable test accuracy after 60 rounds' training.

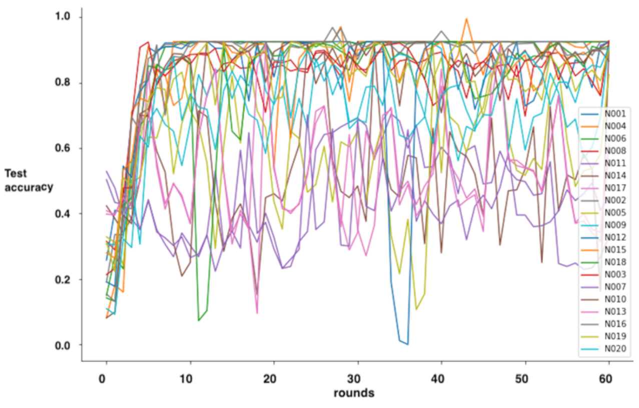
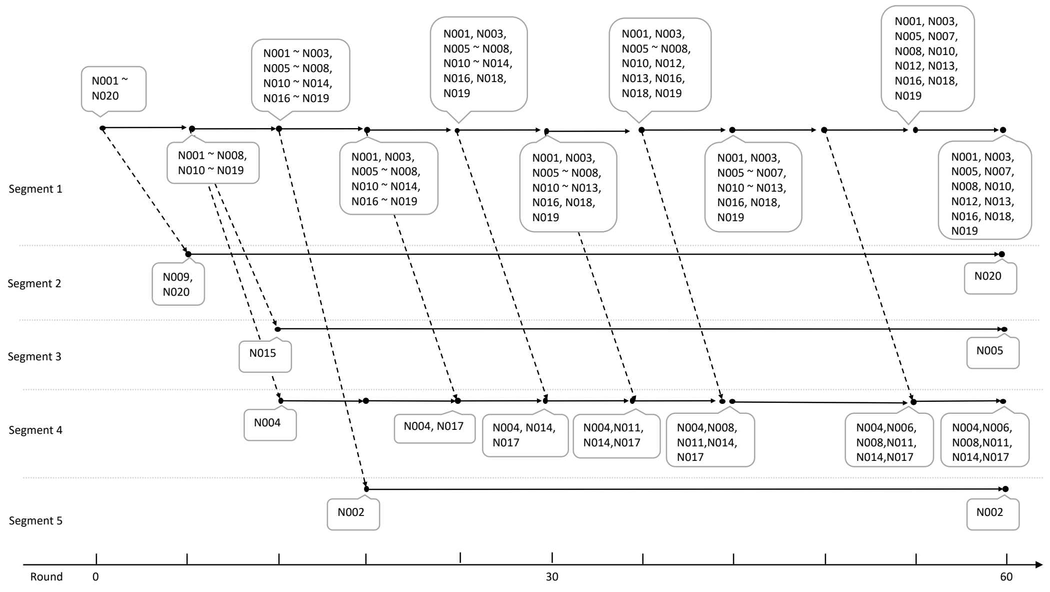
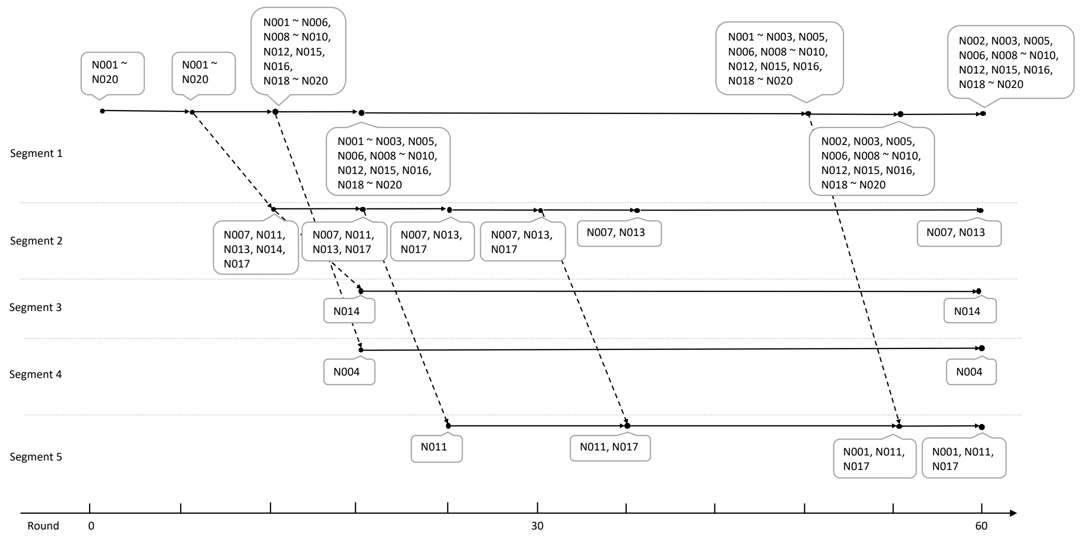


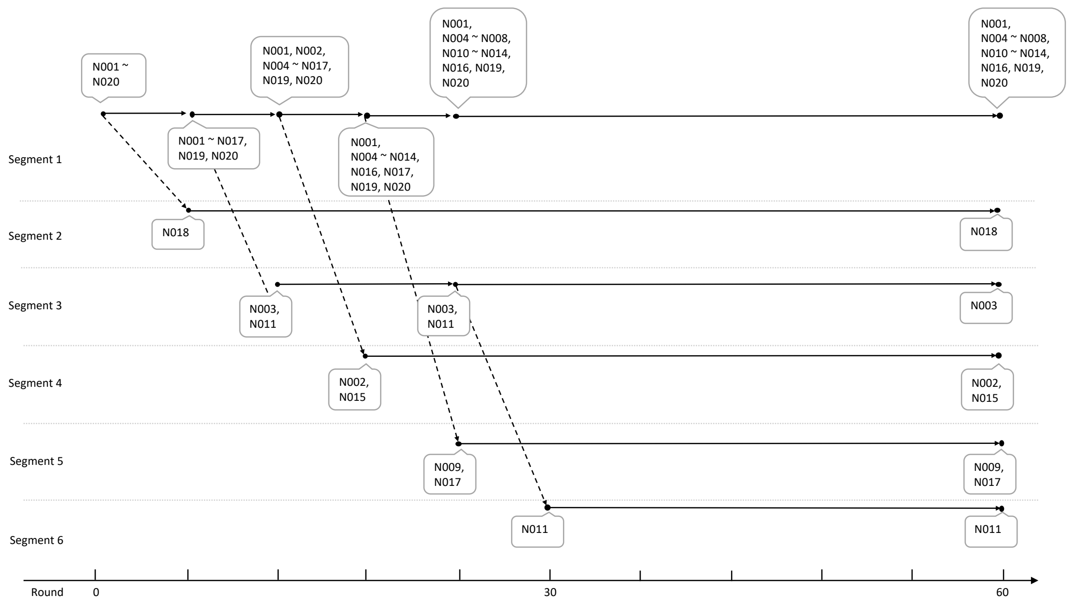
Fig. 8. Test accuracy with Knowledge B: Several participants' local models show an unstable test accuracy during the whole course of training; while the others achieve a stable test accuracy after 60 rounds' training.


N001 ~ N020
N001 ~ N003, N005 ~ N008, N010 ~ N014, N016 ~ N019
N001, N003, N005 ~ N008, N010 ~ N014, N016, N018, N019
N001, N003, N005 ~ N008, N010, N012, N013, N016, N018, N019
N001, N003, N005, N007, N008, N010, N012, N013, N016, N018, N019
N001 ~ N008, N010 ~ N019
N001, N003, N005 ~ N008, N010 ~ N014, N016 ~ N019
N001, N003, N005 ~ N008, N010 ~ N013, N016, N018, N019
N001, N003, N005 ~ N007, N010 ~ N013, N016, N018, N019
N001, N003, N005, N007, N008, N010, N012, N013, N016, N018, N019
Segment 1
Segment 2
N009, N020
N020
Segment 3
N015
N005
Segment 4
N004
N004, N017
N004, N014, N017
N004,N011, N014,N017
N004,N008, N011,N014, N017
N004,N006, N008,N011, N014,N017
N004,N006, N008,N011, N014,N017
Segment 5
N002
N002
Round
0
30
60

(a) The segmentation of the model with Knowledge A as the labeling method.


N001 ~ N020
N001 ~ N020
N001 ~ N006, N008 ~ N010, N012, N015, N016, N018 ~ N020
N001 ~ N003, N005, N006, N008 ~ N010, N012, N015, N016, N018 ~ N020
N002, N003, N005, N006, N008 ~ N010, N012, N015, N016, N018 ~ N020
Segment 1
N001 ~ N003, N005, N006, N008 ~ N010, N012, N015, N016, N018 ~ N020
N002, N003, N005, N006, N008 ~ N010, N012, N015, N016, N018 ~ N020
Segment 2
N007, N011, N013, N014, N017
N007, N011, N013, N017
N007, N013, N017
N007, N013, N017
N007, N013
N007, N013
Segment 3
N014
N014
Segment 4
N004
N004
Segment 5
N011
N011, N017
N001, N011, N017
N001, N011, N017
Round
0
30
60

(b) The segmentation of the model with Knowledge B as the labeling method.



(c) The segmentation of the model with Knowledge C as the labeling method.

Fig. 10. The segmentation of the learning model with different knowledge-based labeling methods. With the progress of the segmented federated learning, 20 participants are divided into several segments, each of which has an independent global model.

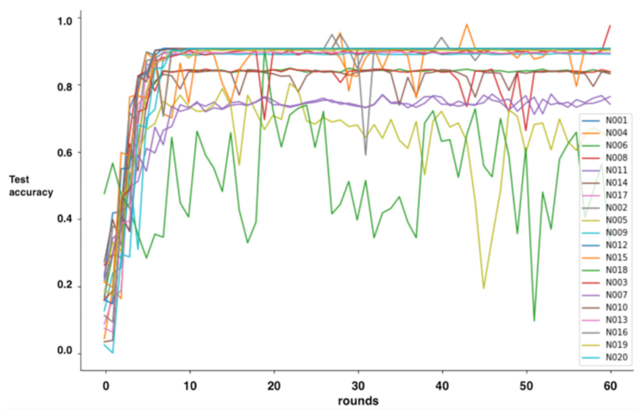


Fig. 9. Test accuracy with Knowledge C: Almost all participants' local models achieve a stable test accuracy after 60 rounds' training.

From the graphs, we can see most participants kept a stable learning progress with Knowledge A and Knowledge C, while several participants gave relatively low accuracy with Knowledge B, however 14 participants among 20 achieved more than 0.800 accuracy. At last, we compute the average validation accuracies of all participants with three different types of knowledge-based labeling methods, and achieve the results as 0.923, 0.813 and 0.877 individually for the intrusion detection.

Furthermore, we visualize the progresses of the learning model's segmentation with three types of labeling methods and give out the final segments of participants in the segmented federated learning scheme (Fig. 10).

V. Conclusion

In this research, a segmented federated learning is adopted to solve the problem of various adaptivity of participants' network to the global model at the central server as well as privacy issues of participants' local data. We use traffic data of 60 days from 20 participants in the LAN-security Monitoring Project. Then we generate the feature maps based on nine types of protocols' communication frequency. Moreover, three types of knowledge-based methods are used for labeling.

Furthermore, a four-layers CNN model is adopted as local machine learning models as well as the global model. We adopt a learning rate of 0.00001, a training batch size of 200, and an epoch of one, conducting the learning for a total sixty rounds. For every six rounds, performance evaluation is conducted, and according to the results, the structure of the learning model is adjusted automatically. It shows a good performance when it comes to the task of intrusion detection in LANs with the segmented federated learning. For future work, a discussion on various training parameters' influence on the precision and stability of the learning model is considered, such as the epoch, the learning rate, and the batch size.

References

[1] "LAN-Security Monitoring Project", www.lan-security.net, Accessed 5th Jan. 2020

[2] Nilamadhab Mishra, Sarojananda Mishra, Support Vector Machine Used in Network Intrusion Detection, National Workshop on Internet of Things (IoT), 2018

[3] T. Omrani, A. Dallali, B. C. Rhaimi and J. Fattahi, Fusion of ANN and SVM classifiers for network attack detection, 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, 2017

[4] M.A. Salama, H.F. Eid, R.A. Ramadan, A. Darwish, A.E. Hassanien, Hybrid intelligent intrusion detection scheme, Soft Computing in Industrial Applications, pp.293–303, 2011

[5] Dhanabal, L. and S. P. Shantharajah, A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms, 2015

[6] J. Yang, J. Deng, S. Li, Y. Hao, Improved traffic detection with support vector machine based on restricted boltzmann machine, Soft Computing 21(11) (2017) 3101–3112, 2017

[7] P.H. Duy, N.N. Diep, Intrusion detection using deep neural network, Southeast Asian Journal of Sciences 5 (2) (2017) 111–125, 2017

[8] J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, Proceedings of the 10th International Conference on Malicious and Unwanted Software (MALWARE), 2015

[9] M. Yousefi-Azar, V. Varadharajan, L. Hamey, U. Tupakula, Autoencoder-based feature learning for cyber security applications, Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), 2017

[10] Briland Hitaj, Giuseppe Ateniese, Fernando Perez-Cruz, Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning, Session C3: Machine Learning Privacy, CCS'17, 2017

[11] Abeshu and N. Chilamkurti, Deep learning: the frontier for distributed attack detection in fog-to-things computing, IEEE Communications Magazine, vol. 56, no. 2, pp. 169–175, 2018

[12] T. D. Nguyen, S. Marchal, M. Miettinen, M. H. Dang, N. Asokan, and A.-R. Sadeghi, A crowdsourced self-learning approach for detecting compromised IoT devices, arXiv preprint, arXiv:1804.07474, 2018

[13] H. B. McMahan, E. Moore, Daniel Ramage, and B. A. Arcas, Federated Learning of Deep Networks using Model Averaging, arXiv preprint, arXiv:1602.05629v1, 2016

[14] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo, Cyber-Physical Systems Security – A Survey, arXiv preprint, arXiv: 1701.04525v1, 2017

[15] Ian Goodfellow, Yoshua Bengio, Aaron Courvile, Deep Learning (Adaptive Computation and Machine Learning), Francis Bach, The MIT Press, 2016

[16] Eric Krokos, Alexander Rowden, Kirsten Whitley, and Amitabh Warshney, Visual Analytics for Root DNS Data, IEEE Symposium on Visualization for Cyber Security, 2018

[17] Anna L. Buczak and Erhan Guven, A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection, IEEE Communications Surveys & Tutorials, Vol. 18, No. 2, 2016

[18] A. Elsaeidy, K. S. Munasinghe, D. Sharma, and Abbas Jamalipour, Intrusion detection in smart cities using Restricted Boltzmann Machines, Journal of Network and Computer Applications 135 76–83, 2019