IoT Malicious Traffic Detection Based on Federated Learning

Yi Shen[1], Yuhan Zhang[2], Yuwei Li[1(✉)], Wanmeng Ding[1], Miao Hu[1], Yang Li[1], Cheng Huang[2], and Jie Wang[1]

[1] College of Electronic Engineering, National University of Defense Technology, Hefei, Anhui, China
liyuwei@nudt.edu.cn

[2] School of Cyber Science and Engineering, Sichuan University, Chengdu, Sichuan, China

Abstract. Nowadays, a large number of IoT devices are manufactured and used in daily life. However, the lack of uniform protocols and standards for IoT devices brings many security risks. Malicious attacks on IoT devices such as Mirai are on the rise, leading to more IoT devices joining botnets and launching DDoS attacks. Therefore, it is necessary to detect malicious traffic of IoT devices. To solve this problem, we propose FLIMT, a federated learning based malicious traffic detection framework for IoT devices. We motivated by the fact that it is not practical to centralize and detect the traffic data sent by IoT devices. Besides, considering the data security and confidentiality standards, it is improper to aggregate data from individual IoT devices into a central computing cluster. FLIMT consists of several GRU-based local detection clients and a central server, where local clients rely on local data for model training and testing, and the central server for model aggregation. The experimental results show that FlIMT achieves high detection accuracy on real data collected from IoT devices, and significantly lessens communication rounds.

Keywords: Internet of Things · Federated Learning · Malicious Traffic

1 Introduction

Along with the maturity of communication technologies, the IoT devices are getting more popular. According to statistics and forecasts, about 64 billion more IoT devices will be deployed and applied in 2025 [1]. At the same time, numerous businesses are rapidly adopting AI-driven solutions. IoT devices offer a new approach to collecting, tracking, and analyzing data that is currently underutilized. The microservice architecture that IoT devices have and the emerging edge-side computing make it viable to serve as a carrier for machine learning. By integrating machine learning technologies into micro-compatible hardware architectures, the IoT devices can be converted to provide AI-enabled services.

S. Goel and P. R. Nunes de Souza (Eds.): ICDF2C 2023, LNICST 570, pp. 249–263, 2024.
https://doi.org/10.1007/978-3-031-56580-9_15

Although IoT can deliver a variety of critical services, it needs to consume as little energy as possible renders its microarchitectural style unsuitable for building computationally expensive security firewalls. Its ability to deal with cyber threats is limited; as Kolias, Constantinos, et al. [4] mentioned, Mirai and other malware can exploit vulnerabilities in IoT devices to gain control of the device and use it to launch additional attacks. Since 2019, IoT malware attacks have surged by 700%, with BASHLITE and Mirai accounting for 97% of IoT malware intercepted [5]. One technique for detecting malware-infected devices is to monitor and analyze the IoT device's behavioral fingerprints such as network traffic. Machine learning and deep learning are already frequently utilized in detecting malicious network traffic. However, they still have a lot of issues. For example, the amount of data from individual devices is insufficient to allow deep learning; more information must be aggregated; and IoT device heterogeneity and computational performance limitations make implementing machine learning-based fraudulent traffic identification a difficult operation [7,8], which also implies that a central entity is required to collect data from different IoT devices and train a global model, followed by distributing the model among clients, or clients provide real-time data to the central entity for detection. However, with the introduction of 5G networks, the volume of data generated by IoT devices is rapidly increasing. The rising expense of communication makes us wonder if such an approach can be improved. Furthermore, this strategy is not applicable to circumstances involving confidential or private data, and we still face significant security risks when the raw data is delivered to the central cluster. Thus, how to identify malicious traffic with low network overhead for heterogeneous IoT devices and interaction networks is of paramount importance.

In such a context, federated learning (FL) [9] is utilized as a strongly adaptive strategy in which the training of each machine learning model is done at the edge of decentralized clients using their own data [10]. The models' weights are then passed to the central entity for aggregation, peer-to-peer transmission, assembly, and fusion to create unique global models for distribution and multiple iterations, avoiding the security issues that the original data faced during transmission and reducing communication resources consumption.

In this paper, we propose a dynamic weight-based federated learning framework FLIMT for detecting malware-infected IoT devices in a real-world IoT traffic environment, allowing for deep learning training at edge clients while avoiding the security issues associated with uploading raw traffic data. We use a Gate Recurrent Unit (GRU) neural network model to build edge clients, divide the N-BaIoT [19] dataset according to different devices and deploy them to each edge client for training the model to detect malicious traffic with accuracy better than the FedAvg, improve the accuracy of different client models for detecting heterogeneous IoT devices, and optimize the FedAvg algorithm to reducing the number of communication rounds while guaranteeing the performance of the detection model. Our contributions in this work can be summarized as follows:

– We designed FLIMT, a malicious traffic detection framework based on improved federated learning and GRU algorithm. Model testing and train-

ing are carried out locally by GRU-based clients, and the models from each client are aggregated and distributed by the central server.
– We innovatively introduced a dynamic weighting mechanism in the aggregation process of the federated learning model. In each aggregation process, the corresponding weights are generated according to the degree of change of the clients model, which effectively improves the efficiency of the model and reduces the number of communication rounds.
– FLIMT achieves excellent results in the real IoT traffic environment. Compared to traditional centralized training and FedAvg, FLIMT achieves the highest accuracy and the best stability on other devices data not participate in previous experiments.

2 Related Work

2.1 Non-federated Method

Constantinos et al. [4] mentioned, the notorious botnet Mirai, and its malware variants have compromised thousands of IoT devices, exploiting them as clients to launch huge distributed denial-of-service attacks against a large number of prominent websites. Alshamkhany, Mustafa, et al. [14] used four typical machine learning models, plain Bayesian, K-nearest neighbor, support vector machine, and decision tree to classify the Bot-IoT [15], UNSW-NB15 [16] dataset for malicious traffic detection and achieved higher accuracy compared to previous work. While machine learning methods are demanding in feature extraction, the difference in detection results between different datasets can be large. Therefore, an increasing number of researchers are applying deep learning in this area. Biswas, Rajib, and Sambuddha Roy [17] used GRU, ANN, and other neural networks for malicious traffic detection on the Bot-IoT [15] dataset and achieved higher accuracy in this dataset. Lucid [18] is a DDoS detection architecture based on a CNN neural network with low processing overhead, short attack detection time, and 98% accuracy, validated on several datasets. Still, it can only detect whether the traffic is a DDoS attack and cannot accurately determine the malware launching the attack. Yair Meidan et al. [19] innovatively proposed using a deep automatic coding machine to identify and evaluate anomalous network traffic, using two malware infecting nine real commercial IoT devices to obtain real traffic for detection. However, the FPR of its detection effect is not stable.

2.2 Federated Method

McMahan et al. [9] first proposed the concept of federated learning, which can be divided into three types, horizontal federated learning, vertical federated learning, and migration federated learning; the most commonly used one is horizontal federated learning, which operates through a distributed architecture, including a central server and multiple working clients, and the data of each client is trained locally. After the local training is completed, the client separately

uploads the model parameters to the server, which aggregates the parameters using the FedAvg algorithm and returns the model parameters to the client, which updates the model for the next round of training. However, the FedAvg algorithm directly weighted the average of the model parameters, which can have a negative impact on the model performance in some circumstances and significantly raise the communication burden. Hao Wang et al. [20] designed The federal learning Favor framework based on reinforcement learning, which does not select all clients at each communication, reduces the bias caused by non-IID data, and effectively reduces the number of communication rounds. However, if the data distribution in the network is imbalanced, the model's accuracy suffers dramatically. Shiqiang Wang et al. [21] came up with a control algorithm that determines the best aggregation parameters to minimize the loss function for a given resource budget, showing better performance under different models and data sets. Felix Sattler et al. [22] proposed a compression method and proposed a new sparse ternary compression (STC) framework that improves the efficiency of federated learning communication and reduces the size of communication messages.

In recent years, Federated learning has also received much attention for IoT malicious traffic detection [23–27]. Nguyen et al. [24] designed DÏoT, a distributed IoT device detection system based on federated learning, for detecting Mirai malware-infected IoT devices in IoT intelligent home networks, DÏoT consists of a security gateway and an IoT security service. Li et al. [25] propose DeepFed, a federated learning-based intrusion detection framework that uses a combination of Convolutional Neural Network (CNN) and Gate Recurrent Unit (GRU). Zhang et al. [26] proposed FLDDoS, which uses a self-encoder-based RNN classification model combined with federated learning for identifying DDoS attack traffic. Still, their work is limited to detecting DDoS attacks. Man et al. [27] proposed an intelligent intrusion detection mechanism FedACNN, which uses a federated learning mechanism to assist the deep learning model CNN to complete the intrusion detection task.

3 Methodology

In this section, we first describe the local neural network model built based on GRU, and then introduce the data preprocessing method. The latest federation learning algorithm is also presented. Eventually, we present FLIMT, a federated learning framework with dynamic weights for detecting malicious IoT traffic.

3.1 GRU-Based Local Model Design

Inspired by [26,31], we design a unidirectional two-layer GRU based as a local model for IoT malicious traffic detection. The local model is constructed with the same hyperparameters at each client and uses its own local data for model training and testing. This section introduces the principle of the local model and the design concepts.

The Gated Recurrent Unit (GRU) [28] is a variant of the cyclic neural network RNN. The structure of the GRU input and output is similar to that of the traditional RNN. The GRU performs similarly to the LSTM in many cases, but GRU reduces the computational complexity compared to LSTM. It can effectively improve computational efficiency.

GRU input contains x_t, and a hidden state h_{t-1} is handed down from the previous node, which contains the relevant information of the previous node. After the reset gate and update gate operations, the new hidden state h_t is ultimately output. r stands for the reset gate signal, and z for the update gate signal. First, apply function *sigma* to calculate the reset gate signal via (1) to achieve information forgetting during the reset operation. After receiving the reset gate signal, use (2) to obtain the data $h_{t-1}^{'}$ after reset, and splice the $h_{t-1}^{'}$ with the input x_t. Deflate data to -1 and 1 by tanh. Obtain $h^{'}$, which represents the state of remembering the current moment, as indicated in (3).

$$r = \sigma(W_r[h_{t-1}, x_t]) \tag{1}$$

$$h_{t-1}^{'} = h_{t-1} \odot r \tag{2}$$

$$h^{'} = tanh(W[h_{t-1}^{'}], x_t) \tag{3}$$

In the update operation, two steps of forgetting and remembering are realized. Firstly, the update gating signal σ function is calculated by (4), and the update of the hidden state is completed by (5).

$$z = \sigma(W_z[h_{t-1}, x_t]) \tag{4}$$

$$h_t = (1 - z) \odot h_{t-1} + z \odot h^{'} \tag{5}$$

The network structure has a considerable impact on the detection results when using neural network models for detection tasks. However, most edge devices lack memory and computing capabilities, making it impossible to store and execute complicated GRU models. Therefore, the structure of our GRU model is relatively simple while ensuring accuracy. The input layer of our model receives data as 115-dimensional traffic data features after data preprocessing, and in the hidden layer, two GRUs are stacked together to form a stacked GRU layer; the output is received by the fully connected layer, which then uses Relu for nonlinear variation, and the final layer is output through the fully connected layer, which outputs a classification of different malicious and normal traffic.

3.2 Data Preprocessing

In real application scenarios, how to obtain data from real IoT devices that can be used to detect malicious traffic must also be considered. When an IoT device sends a data packet, a behavioral snapshot of the host that sent the packet communication and its communication protocol is taken. Different information such as packet size and the number of packets are extracted to generate 23 features. The snapshot is extracted by extracting feature sets of the same shape

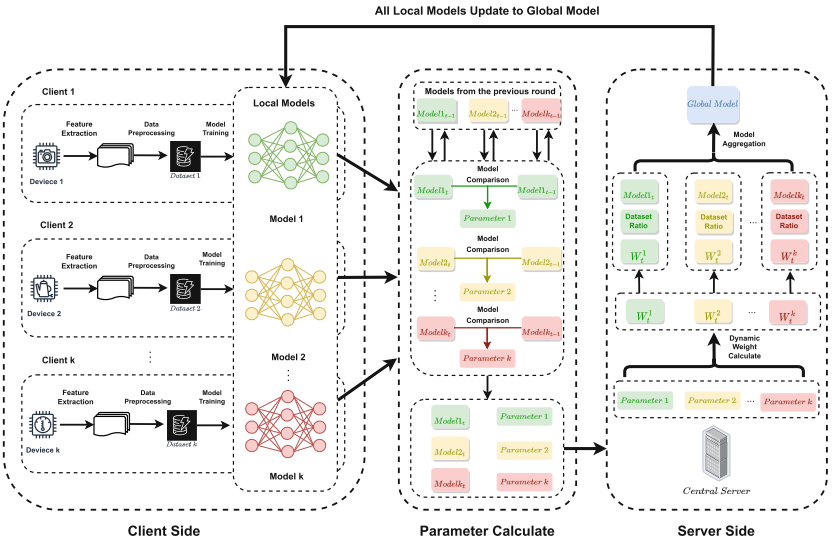


Fig. 1. The framework of FLIMT.

in 5-time windows (100 ms, 500 ms, 1.5 s, 10 s, and 1 min). Get the context of the data packet, then generate a data sample with 115 features and store it. After the dataset is acquired and collected, it is classified and stored according to device and traffic type.

After the feature extraction is done and the dataset is generated, we need to process it in order to make it usable by the neural network. First, pad all NaN values with zeros, and treat string types as numeric types. Finally, we standardize all features by (6) to improve the classification accuracy.

$$x = \frac{x - \bar{x}}{\sigma} \tag{6}$$

3.3 Federated Learning

Horizontal federated learning is also known as feature-aligned federated learning; that is, the data features of clients participating in horizontal federated learning are aligned, and the data features of different clients overlap more while the sample ID overlaps less. In our experiments, the characteristics of the datasets used in the experiments are fixed, and horizontal federated learning is the best choice. We present a federated learning framework for IoT malicious traffic detection, which comprises a central server and several clients in a horizontal federated learning architecture. The key concept of federated learning is data localization and model transmission changes. Specifically, the central server initializes the global model before training starts and sends the model to each client. The model is trained through local data to obtain a new local model. After a period

of training, the client uploads the new local model parameters to the central server. The central server aggregates the model parameters of multiple clients in a specific way as a new global model parameter and sends the latest global model parameters to the client; the client updates the local model, then uses the local data to train the local model again, uploads the local model parameters, the central server aggregates the parameters again, updates the global model parameters. The process is repeated until all training is completed. In particular, in the architecture we designed, in each communication that the client uploads local model parameters to the central server, we will upload an additional parameter to calculate the weight of each local model's influence on updating the global model (we will discuss in detail later).

McMahan et al. [9] proposed the federated learning aggregation algorithm FedAvg, which solves the problem that the fusion of FedSGD models takes too long. When compared to the synchronous stochastic gradient descent algorithm, communication cost is a large constraint in the FL situation; their FedAVG may greatly minimize the number of communication cycles, and FedAvg primarily includes the following main parameters, C represents the ratio of clients performing the calculation in each round, K represents the total number of clients, B is the batch size for client training, E is the number of client training epochs, and η indicates the learning rate. On the server-side, first, initialize the global model parameters w_0. For each round of communication, m clients are randomly selected by (7). Each client k receives the current global model parameter w_t and is instructed to perform model training and update model parameters. After the training is completed, the model parameters are updated from w_t^k to w_{t+1}^k. The model parameters are uploaded to the server, and the server integrates all w_{t+1}^k to obtain new global model parameters via (8), where n represents the total number of samples, and n_k represents the number of client samples.

$$m = max(C \cdot K, 1) \tag{7}$$

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{8}$$

For each client selected in a round of communication, divide its data set into batches according to the parameters B, each batch size is B, for each batch of data b, E rounds update is needed. Calculate the loss gradient of the data block by (9), and update the gradient descent to obtain a new local model parameter w. Following the completion of all calculations by this client, the final local model parameters w are transferred to the server-side for aggregation as w_{t+1}^k.

$$w = w - \eta \nabla \iota(w; b) \tag{9}$$

3.4 Improved Federated Learning for Detection

We note that in the FedAvg algorithm, the global model is updated by aggregation using (8), and this aggregation uses a simple weighted average technique that only considers the data distribution without taking into account the impact

of other client-side factors on the global model. On this basis, we propose a dynamic weighting mechanism based on the degree of client-side model changes. After each client completes a communication round of training, it evaluates the changes of the new model parameters with those of the previous round. It sends the evaluation results and the new model parameters to the server, which assesses the change of each client and generates the corresponding weights. The lower the degree of client model change, the better the client fits the global model, and the higher the weight. The addition of dynamic weights accelerates the convergence of the global model and reduces the impact of clients that do not fit the global model. In Algorithm 1, we describe FLIMT in detail. The architecture of FLIMT is shown in Fig. 1.

Before the first communication round begins, the global model parameters are initialized and sent down like FedAvg, m clients are randomly selected for this communication round, and each of these clients k gets initialized local model parameters w_0^k. Following that, After each communication round t, that is, after the client receives the new global model and executes training updates, it obtains new local model parameters w_t^k. The latest local model parameters are subtracted from the matrix of d neural network parameters corresponding to the previous round of model parameters to obtain the difference matrix $(D_t)_i$ respectively. The sum h_t^k of the two-norms of all difference matrices are received by (11) as a benchmark for the variation of model parameters.

$$(D_t)_i = (w_t^k)_i - (w_{t-1}^k)_i \tag{10}$$

$$h_t^k = \sum_{i=1}^{d} \sqrt{\lambda_{max}((D_t)_i^T (D_t)_i)} \tag{11}$$

The client uploads the latest model parameters w_t^k together with the model change metric h_t^k to the server-side, which receives the uploads from all m clients, utilizes h_t^k for weight calculation. Because of the substantial variations in model parameters, first normalize h_t^k to $h_t^{k'}$ using function sigmoid. Then the corresponding dynamic weights a_t^k for each client in this round are obtained by (14). After the calculation is completed, the dynamic weights are referenced and aggregated using (15) to obtain the new global model parameters w_t and enter the next communication cycle.

$$f_\sigma(x) = \frac{1}{1 + e^{-x}} \tag{12}$$

$$h_t^{k'} = \frac{1}{f_\sigma(h_t^k)} \tag{13}$$

$$a_t^k = \frac{m * h_t^{k'}}{\sum_{k=1}^{m} h_t^{k'}} \tag{14}$$

$$w_t = \sum_{k=1}^{m} a_t^k \frac{n_k}{n} w_t^k \tag{15}$$

Algorithm 1. FLIMT with Dynamic Weights

procedure server-side:
  initialize w_0
  for each round t=1,2,3... **do**
    $m \leftarrow max(C \cdot K, 1)$
    $S_t \leftarrow$(random set of m clients)
    for each client $k \in S_t$ **in parallel do**
      $w_t^k, h_t^k \leftarrow$**ClientUpdate**(k, w_{t-1})
      $h_t^{k'} \leftarrow \frac{1}{f_\sigma(h_t^k)}$
      $a_t^k \leftarrow \frac{m * h_t^{k'}}{\sum_{k=1}^{m} h_t^{k'}}$
    end for
    $w_t \leftarrow \sum_{k=1}^{m} a_t^k \frac{n_k}{n} w_t^k$
  end for
end procedure
procedure clientupdate(k, w)*Run on client* n
  $B \leftarrow$(Split local Client data into batches of size B)
  for each local epoch e from 1 to E **do**
    for each client $k \in S_t$ **in parallel do**
      $w_t^k \leftarrow w - \eta \nabla \iota(w; b)$
    end for
    $(D_t)_i \leftarrow (w_t^k)_i - (w_{t-1}^k)_i$
    $h_t^k \leftarrow \sum_{i=1}^{d} \sqrt{\lambda_{max}((D_t)_i^T (D_t)_i)}$
  end for
  return w_t^k, h_t^k
end procedure

When model parameters aggregation is conducted in FedAvg, the client models that were not selected for an update in this round also participate in the global model aggregation, which is optional. Since these clients do not have dynamic weights in that round, we choose the more common strategy in real applications, which is to aggregate only those updated clients model parameters.

4 Evaluation

To verify the performance and effectiveness of FLIMT, we train and validate the framework based on the dataset presented in this section and the evaluation metrics and compare FLIMT with non-federal learning, FedAvg algorithm, etc. The server used for this experiment is Ubuntu 20.04.4 OS, the processor is Intel(R) Xeon(R) CPU @ 2.00 GHz, and the GPU is NVIDIA Corporation GP100GL [Tesla P100 PCIe 16 GB]. We used Pytorch, a deep learning library for Python, for model construction and simulation of the federated learning system.

4.1 Dataset

Datasets that can effectively reflect the real world are necessary. In our experiments, we selected N-BaIoT [19] as the dataset, which addresses the lack of

Table 1. Number of samples for each device.

Devive	Benign	Mirai	BASHLITE
Danmini_Doorbell	49548	652100	316650
Ecobee_Thermostat	13113	512133	310630
Ennio_Doorbell	39100	0	316400
Philips_B120N10_Baby_Monitor	175240	610714	312723
Provision_PT_737E_Security_Camera	62154	436010	330096
Provision_PT_838_Security_Camera	98514	429337	309040
Samsung_SNH_1011_N_Webcam	52150	0	323072
SimpleHome_XCS7_1002_WHT_Security_Camera	46585	513248	303223
SimpleHome_XCS7_100_WHT_Security_Camera	19528	514860	316438
Total	**555932**	**3668402**	**2838272**
Proportion	**7.87%**	**51.94%**	**40.19%**

public botnet datasets, especially in the context of IoT. It presents accurate traffic data collected from nine distinct types of commercial IoT devices authentically infected by Mirai and BASHLITE, as well as normal traffic data generated by these devices. The dataset has been segmented by device, which is consistent with the federated learning concept. Each sample in the dataset corresponds to a network packet that Wireshark has sniffed. For each packet, 115 numerical features characterizing the packet context are extracted. The available features are statistics about the size, count, and jitter of aggregated network packets over the past 100 ms, 500 ms, 1.5 s, 10 s, and 1 min. In Table 1, we record the number of benign and attack samples, as well as the total number from all devices. Before starting the experiments, we have preprocessed the dataset by dividing it into devices to simulate distributed scenarios. The data were standardized and separated into a training set and a testing set for local model training and testing according to the ratio of 7.5:2.5, respectively.

4.2 Model Construction

We build three classification models typically used in malicious traffic detection, train them locally, and evaluate their performances. They are MLP, RNN, and GRU, respectively. We use Adam as the optimizer, CrossEntropyLoss as the loss function, and set the learning rate $\eta = 1e{-}4$; we randomly select a device's data from the dataset to train and test the model respectively and develop the iteration round $e = 50$. We tested the hyperparameters of the three models, and the optimal configuration is as follows. The MLP consists of three hidden layers with 256, 128, and 3 hidden nodes, respectively. The RNN consists of a unidirectional two-layer RNN layer with 330 hidden nodes, connected with two fully-connected layers for dimensional transformation, and uses the ReLu function for nonlinear transformation between the fully-connected layers. The

structure of GRU is similar to that of RNN, except that a GRU layer replaces the RNN layer. To better evaluate the model performance, we use the testing set for evaluation and get the corresponding results as shown in Table 2, GRU works best in local training, so it is applied to the subsequent federated learning model.

Table 2. Local model evaluation.

Model	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
MLP	99.45	99.48	99.58	99.53
RNN	99.92	99.90	99.91	99.91
GRU	**99.95**	**99.95**	**99.93**	**99.94**

Table 3. Accuracy of different data types.

Model	Benign	Mirai	BASHLITE
Centralized	99.94	99.96	99.95
FedAvg	99.93	99.95	99.93
FLIMT	**99.94**	**99.99**	**99.96**

Table 4. Evaluation results of different models.

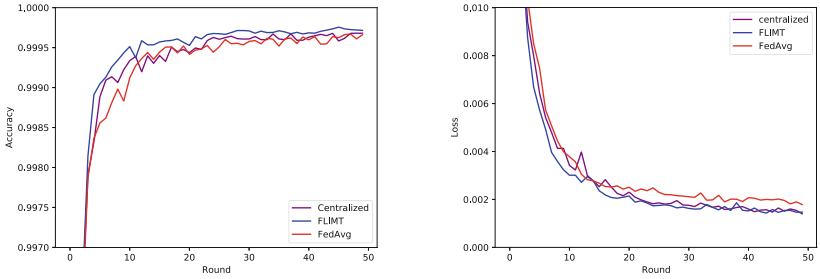
Model	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
Centralized	99.95	99.96	99.96	99.96
FedAvg	99.95	99.94	99.95	99.95
FLIMT	**99.98**	**99.98**	**99.98**	**99.98**

4.3 Federated Method

We use FedAvg and centralized training to compare with our FLIMT framework. For the two federated learning models, their hyperparameters are set to $B = 64$, $E = 1$, $K = 5$, $C = 1$, and $\eta = 1e{-}4$. In other words, model training is performed on five local clients, and both the evaluation and aggregation of the model are performed after each epoch of training. The GRU local model is initialized at each local client, and each local client has a processed dataset of different devices (except Ennio_Doorbell and Samsung_SNH_1011_N_Webcam due to the lack of Mirai data traffic). For traditional centralized training, 50 training epochs are set, and the model evaluation is performed after each epoch. The data used by the federated learning model is mixed for the centralized model. Among them, the evaluation result of the federated learning model is recorded by taking the average value of all clients. As can be seen in Fig. 2(a) and Fig. 2(b), under the

Table 5. Accuracy with different K values.

Model	K = 3	K = 4	K = 5	K = 6
Centralized	99.96	99.96	99.95	99.96
FedAvg	99.94	99.94	99.95	99.95
FLIMT	**99.97**	**99.98**	**99.98**	**99.95**



(a) Accuracy of Three Models (b) Loss of Three Models

Fig. 2. The experimental results.

same data set, after 50 rounds of communication, FLIMT achieved the highest accuracy rate of 99.98% and stabilized the loss value around 0.001; additionally, the convergence speed is faster and 20% fewer communication cycles are needed when compared to the FedAvg.

Experiments show that FLIMT can achieve high accuracy while preserving data privacy. In Table 3, we record the performance of different methods for different traffic detection. It can be seen that FLIMT has the best performance in identifying two kinds of malicious traffic, and the traffic recognition rate for Mirai-infected devices reaches 99.99%. Comparable to FedAvg and traditional centralized models in identifying normal traffic. We also recorded the indicators of the three methods in Table 4, and it can be seen that FLIMT is still the best. We also made several changes to the value of K to verify the performance of our framework under different numbers of clients; the results can be seen in Table 5; FLIMT still performs well.

We also validate the effectiveness of our approach for unknown devices, where the data traffic information characteristics may vary between different IoT devices. For devices that are not involved in federated learning, which is also something that needs to be taken into account in the federated learning architecture. We selected the data sample from the device Danmin_Doorbell, which was not involved in any training or testing. It contains a more balanced ratio of normal to abnormal traffic, and it is not of the same type or the same manufacturer as the rest of the IoT devices. We randomly select 25% of the data samples from this device as the testing set, replace the testing set we previously

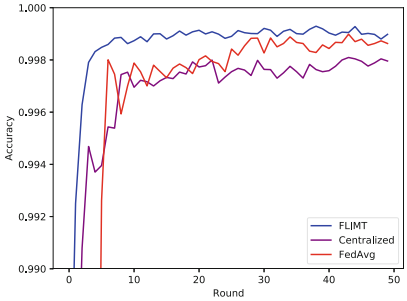


Fig. 3. Evaluation accuracy of unknown device.

used in this client, reinitialize the model and follow the same steps to train and test the model.

We can obtain the test accuracy results as shown in Fig. 3, where we can see that all the models' accuracy decreases for the data samples of unknown devices. In contrast, FLIMT reduces to 99.89%, while centralized learning and FedAvg decrease to 99.78% and 99.85%. Moreover, our proposed method was the least affected, while centralized learning was the most affected. It can also be seen that the centralized learning and FedAvg accuracy curves are more volatile, and our more proposed method is significantly less volatile and more stable. We can conclude that our proposed method has better detection for malicious traffic generated by unknown IoT devices, and the model generalization ability is better.

5 Conclusion

In this paper, we propose FLIMT, a federated learning-based framework for malicious traffic detection in IoT devices, which can use data from multiple clients to collaboratively optimize the malicious traffic detection model while protecting data privacy. Meanwhile, we propose a novel dynamic weighting mechanism, where each client compares the brand new model parameters with the previous round after each round of federated learning communication, obtains the model change parameters, and uploads them to the server, which collects the model change parameters of each client and calculates the weight of that client for this round, which takes effect when federated learning is performed for central model aggregation. After our experiments, our framework achieves the highest detection accuracy of 99.98% and improves stability compared to other methods, reduces the number of federated learning communication rounds needed to achieve high accuracy, and reduces communication resource usage. Predictably, our proposed method may be able to make mitigation in poisoning attacks against federated learning. We intend to investigate more kinds of malicious traffic in the future so that existing models can better match real-world scenarios.

Acknowledgment. This research is funded by the National Natural Science Foundation of China (NSFC) under No. 62202484.

References

1. Riad, K., Huang, T., Ke, L.: A dynamic and hierarchical access control for IoT in multi-authority cloud storage. J. Netw. Comput. Appl. **160**, 102633 (2020)
2. Catarinucci, L., et al.: An IoT-aware architecture for smart healthcare systems. IEEE Internet Things J. **2**(6), 515–526 (2015)
3. Stergiou, C.L., Psannis, K.E., Gupta, B.B.: IoT-based big data secure management in the fog over a 6G wireless network. IEEE Internet Things J. **8**(7), 5164–5171 (2020)
4. Kolias, C., et al.: DDoS in the IoT: mirai and other botnets. Computer **50**(7), 80–84 (2017)
5. Wodecki, N.: Zscaler Study Confirms IoT Devices are a Major Source of Security Compromise, Reinforces Need for Zero Trust Security. Web (2021). https://www.globenewswire.com
6. Elrawy, M.F., Awad, A.I., Hamed, H.F.A.: Intrusion detection systems for IoT-based smart environments: a survey. J. Cloud Comput. **7**(1), 1–20 (2018)
7. Wu, J., et al.: Application-aware consensus management for software-defined intelligent blockchain in IoT. IEEE Netw. **34**(1), 69–75 (2020)
8. Liang, H., et al.: MBID: micro-blockchain-based geographical dynamic intrusion detection for V2X. IEEE Commun. Mag. **57**(10), 77–83 (2019)
9. McMahan, B., et al.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. PMLR (2017)
10. Yang, Q., et al.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. (TIST) **10**(2), 1–19 (2019)
11. Meneghello, F., et al.: IoT: Internet of threats? A survey of practical security vulnerabilities in real IoT devices. IEEE Internet Things J. **6**(5), 8182–8201 (2019)
12. HaddadPajouh, H., et al.: A survey on Internet of Things security: requirements, challenges, and solutions. Internet of Things **14**, 100129 (2021)
13. Mekala, M.S., et al.: Resource offload consolidation based on deep-reinforcement learning approach in cyber-physical systems. IEEE Trans. Emerg. Top. Comput. Intell. **6**(2), 245–254 (2020)
14. Alshamkhany, M., et al.: Botnet attack detection using machine learning. In: 2020 14th International Conference on Innovations in Information Technology (IIT). IEEE (2020)
15. Koroniotis, N., et al.: Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Future Gener. Comput. Syst. **100**, 779–796 (2019)
16. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS). IEEE (2015)
17. Biswas, R., Roy, S.: Botnet traffic identification using neural networks. Multimedia Tools Appl. **80**(16), 24147–24171 (2021)
18. Doriguzzi-Corin, R., et al.: LUCID: a practical, lightweight deep learning solution for DDoS attack detection. IEEE Trans. Netw. Serv. Manag. **17**(2), 876–889 (2020)
19. Meidan, Y., et al.: N-baiot-network-based detection of IoT botnet attacks using deep autoencoders. IEEE Perv. Comput. **17**(3), 12–22 (2018)

20. Wang, H., et al.: Optimizing federated learning on non-IID data with reinforcement learning. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE (2020)
21. Wang, S., et al.: When edge meets learning: adaptive control for resource-constrained distributed machine learning. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE (2018)
22. Sattler, F., et al.: Robust and communication-efficient federated learning from non-IID data. IEEE Trans. Neural Netw. Learn. Syst. **31**(9), 3400–3413 (2019)
23. Rey, V., et al.: Federated learning for malware detection in IoT devices. Comput. Netw. **204**, 108693 (2022)
24. Nguyen, T.D., et al.: DIoT: a federated self-learning anomaly detection system for IoT. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE (2019)
25. Li, B., et al.: DeepFed: federated deep learning for intrusion detection in industrial cyber-physical systems. IEEE Trans. Ind. Inform. **17**(8), 5615–5624 (2020)
26. Zhang, J., et al.: FLDDoS: DDoS attack detection model based on federated learning. In: 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE (2021)
27. Man, D., et al.: Intelligent intrusion detection based on federated learning for edge-assisted Internet of Things. Secur. Commun. Netw. **2021**, 9361348 (2021)
28. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
29. Mirsky, Y., et al.: Kitsune: an ensemble of autoencoders for online network intrusion detection. arXiv preprint arXiv:1802.09089 (2018)
30. Alsaedi, A., et al.: TON_IoT telemetry dataset: a new generation dataset of IoT and IIoT for data-driven intrusion detection systems. IEEE Access **8**, 165130–165150 (2020)
31. Mothukuri, V., et al.: Federated-learning-based anomaly detection for IoT security attacks. IEEE Internet Things J. **9**(4), 2545–2554 (2021)