Federated Learning for Internet of Things

Tuo Zhang*, Chaoyang He*, Tianhao Ma, Lei Gao, Mark Ma, Salman Avestimehr
University of Southern California
Los Angeles, California, USA

ABSTRACT
Federated learning can be a promising solution for enabling IoT cybersecurity (i.e., anomaly detection in the IoT environment) while preserving data privacy and mitigating the high communication/storage overhead (e.g., high-frequency data from time-series sensors) of centralized over-the-cloud approaches. In this paper, to further push forward this direction with a comprehensive study in both algorithm and system design, we build FedIoT platform that contains FedDetect algorithm for on-device anomaly data detection and a system design for realistic evaluation of federated learning on IoT devices. Furthermore, the proposed FedDetect learning framework improves the performance by utilizing a local adaptive optimizer (e.g., Adam) and a cross-round learning rate scheduler. In a network of realistic IoT devices (Raspberry PI), we evaluate FedIoT platform and FedDetect algorithm in both model and system performance. Our results demonstrate the efficacy of federated learning in detecting a wider range of attack types occurred at multiple devices. The system efficiency analysis indicates that both end-to-end training time and memory cost are affordable and promising for resource-constrained IoT devices. The source code is publicly available at https://github.com/FedML-AI/FedIoT

CCS CONCEPTS
• Security and privacy → Distributed systems security.

KEYWORDS
federated learning, cybersecurity, hardware system design

1 INTRODUCTION
Along with the faster Internet speed and more endpoints brought by the 5G, billions of IoT devices online will be deployed [16]. Meanwhile, the traffic volume of IoT-based DDoS attacks reaches unprecedented levels [1]. Consequently, the upgrade for anomaly detection in the IoT field is imminent. The traditional anomaly detection model is trained with centralized data from many IoT devices or solely trained on data from a small scale of devices. However, the centralized over-the-cloud approaches no longer fit the 5G era due to data privacy and extremely high communication/storage overhead (e.g., high-frequency data from time-series sensors) for pooling data from many IoT devices. This is especially the case when the attacks spread to large-scale attack types and devices, since centralizing the data from small-scale devices is difficult to obtain an effective detection model.

Federated Learning (FL) can offer an alternative approach for IoT cybersecurity while mitigating the above challenges. FL is a trending paradigm that can train a global or personalized model without centralizing data from edge devices [5]. There have also been several recent works that bring FL to the realm of cybersecurity. For example, DIoT [12] employs a federated learning approach to anomaly-detection-based intrusion detection in IoT devices. IoTDefender [2] is another similar framework but obtains a personalized model by fine-tuning the global model trained with FL. [4] evaluates FL-based anomaly detection framework with learning tasks such as aggressive driving detection and human activity recognition. [8] further proposes an attention-based CNN-LSTM model to detect anomalies in an FL manner, and reduces the communication cost by using Top-k gradient compression. Recently, [15] even evaluates the impact of malicious clients under the setting of FL-based anomaly detection. However, these works do not evaluate the efficacy of FL in a wide range of attack types and lack system design and performance analysis in a real IoT platform.

To further push forward the research in FL-based IoT cybersecurity, we build FedIoT platform with a simple but effective design philosophy that lays the foundation for future scientific research. The overall design spans dataset, model, algorithm, and system design. More specifically, we propose a federated learning algorithmic framework, FedDetect which utilizes adaptive optimizer (e.g., Adam) and cross-round learning rate scheduler, rather than naive FedAvg [10] for local training. Furthermore, FedDetect supports both global threshold and personalized threshold for different scenarios. In order to verify the effectiveness of FL for IoT security, we design a novel method to synthesize the testset from public dataset for FL-based IoT cybersecurity research. Its design aims to evaluate whether the global model obtained through FL training can recognize more attack types and has higher detection performance (the evaluation and test dataset cover all attack types on the entire IoT network). In addition, we build FedIoT platform for realistic IoT devices with a high degree of modularization and flexible APIs. As such, we can add new data, models, and algorithms with only lightweight modification. Most importantly, FedIoT supports both IoT edge training (e.g., Raspberry PI) and CPU/GPU-based distributed training with the support of MQTT and MPI communication back-end, respectively. We evaluate FedIoT platform and FedDetect algorithm with both global threshold and personalized threshold on N-BaIoT [11] and LANDER [19] dataset, and also analyze the system performance comprehensively (i.e., computational speed, communication cost, and memory cost).

Our results demonstrate the efficacy of federated learning in detecting a large range of attack types. More specially, we find that the global model obtained through FedDetect training has higher detection performance, while only centralizing the data from a few IoT devices has worse performance due to insufficient benign training samples or attack types. FedDetect with personalized threshold also suggests that the detection based on FL may beat solely training on insufficient local data of a single device. Our system efficiency analysis shows that the training time memory cost occupies only a small fraction of the entire host memory of the IoT platform (Raspberry PI), and the end-to-end training time is feasible (less than 1 hour) for practical applications, although the
ratio of communication cost is high. In essence, we summarize our contributions as follows:

- We provide a novel method to synthesize dataset for FL-based IoT cybersecurity research, aiming at evaluating the efficacy of FL in recognizing attack types in a wide range.
- We propose a federated learning framework, FedDetect, for IoT cybersecurity. Most importantly, FedDetect incorporates local adaptivity and cross-round learning rate scheduler for effective distribution optimization and also supports both global and personalized threshold for different scenarios.
- We build FedIoT platform for realistic IoT devices (e.g., Raspberry PI). Our performance analysis, system design, and flexible APIs demonstrate the feasibility and generality for future exploration.

2 ALGORITHM AND SYSTEM DESIGN

2.1 Overview

Federated learning (FL)-based IoT cybersecurity aims to detect network intrusion in IoT devices without centralizing a large amount of high frequent edge data. A generic setting is that many IoT devices collaboratively train a deep Autoencoder model for anomaly detection via federated learning. In this work, our goal is to build a FL system, FedIoT, for this setting to analyze both algorithmic and system performance in real IoT devices (e.g., Raspberry Pi).

We build FedIoT platform with a simple but effective design philosophy that lays the foundation for future scientific research. The overall design is illustrated in Figure 1. More specifically, the entire software architecture consists of three layers: the application layer, the algorithm layer, and the infrastructure layer. We make each layer and module performs its duty and has a high degree of modularization. In the application layer, FedIoT provides a one-line API to launch the federated training on IoT devices in a distributed computing manner. This API takes non-I.I.D. datasets (Section 2.2) and a simple but effective deep Autoencoder model (Section 2.3) as its input; at the algorithm layer, FedIoT supports various FL algorithms such as FedAvg [9], FedOPT [14] and their customized versions FedDetect for anomaly detection (Section 2.4); at the infrastructure layer, FedIoT aims to support lightweight communication APIs with MQTT [3] (i.e., Message Queuing Telemetry Transport, a standard for IoT messaging), and customized PyTorch library [13] that can execute primitives of on-device model training such as forward propagation, calculating loss function and back-propagation (Section A). Our proposed light-weighted FedIoT framework could support the direct implementation of Federated Learning on AI-enabled IoT edge devices, such as Raspberry Pi and NVIDIA Jetson Nano.
2.2 Dataset and Preprocessing

We introduce a widely used public IoT dataset for anomaly detection and then synthesize a dataset for realistic evaluation on the FL-based method. In addition, we also introduce another novel private dataset to consolidate the evaluation.

N-BaIoT dataset [11] is a widely used public dataset for research on anomaly detection of IoT data. N-BaIoT captures network traffic flow from 9 commercial IoT devices authentically attacked by network intrusions. In the original N-BaIoT dataset, it provides 115 statistic features, which could be severely influenced by the hostile attack. Each IoT device has 2 subsets: one benign set containing normal network flow data only, and one attack data subset consisting of two common malware attacks, Mirai and BASHLITE, which each contain five different kinds of attack types.

USC LANDER IoT Operation Traces-20200127 dataset [19] is one of the latest datasets for the research of the operational traffic on IoT edge devices. The LANDER dataset contains 10-day operational traffic for 14 different widely-used IoT devices located in a LAN network without any types of attack. The detailed data distribution including the statistic features of these two datasets will be shown as tables in the Appendix.

In order to verify the effectiveness of FL for IoT security, different from previous works [2, 4, 8, 12, 15], we hope to learn a detection model from benign data widely distributed in different types of devices that can identify a larger range of attacks. Specifically, we hope that the data design meets three characteristics: 1. It contains training data of multiple device types (benign data); 2. Each device has no full set of all attack types; 3. The evaluation and test dataset should cover all attack types on the entire IoT network. These requirements are based on several real-world assumptions:

- From the perspective of benign data, features of the network data flow among different types of devices are inconsistent. For example, a surveillance camera will record in real-time (24x7 hours), while the data generated by a doorbell is intermittent.
- The detection model should have the ability to identify multiple attack types, but the types of attacks encountered by a single device are likely to be only part of the full set. Only by learning with the feature of all attack types in the entire IoT network, the detection model on a single device can have the ability to detect unknown attacks in a wide range.
- Because of privacy (e.g., camera video) and extremely high communication/storage overhead (e.g., high-frequency data from time-series sensors), it is infeasible to centralize data on massive devices.

Therefore, we use N-BaIoT and LANDER to synthesize the testset for FL-based IoT cybersecurity research. Our synthesized testset is generated by the following rules: 1. For each device, we assign the first 2/3 of selected benign data as the training data and the rest 1/3 as the evaluation dataset (i.e., calculating the anomaly threshold, see Section 2.3); 2. We enforce the global test dataset to compose all devices’ benign data and all types of attack data. More specifically, for each device, we randomly select 5000 benign data samples and 5000 malicious data samples from a wide range of attack types (some devices may not have sufficient data samples from dataset).

Intuitively, the global model obtained through FL training can recognize more attack types and have higher detection performance, while the local training alone may have poor performance due to insufficient benign training samples or attack types.

2.3 Anomaly Detection with Deep Autoencoder

We apply Deep Autoencoder [17] as the model for anomaly detection. Deep Autoencoder is simple but effective and does not lose the generality to evaluate FL algorithms and our FedIoT platform. Other advanced models (e.g., Variational Autoencoder, or attention-based CNN-LSTM [8]) can also be applied into our framework without additional engineering efforts (see our system design in Section A).

![Autoencoder Architecture](image)

Figure 2: Autoencoder Architecture

**Model Definition.** Deep Autoencoder focuses on the reconstruction of the input data in an unsupervised learning manner. Figure 2 shows an example of the model architecture. Essentially, Autoencoder splits the neural network into two segments, the encoder $f_{\theta_e}$ and the decoder $f_{\theta_d}$. Encoder $f_{\theta_e}$ compresses the input $x$ to a latent space $z$. Decoder $f_{\theta_d}$ then attempts to restore the original image after some generalized non-linear transformation. Mathematically, the loss function can be written as $\mathcal{L}(x, x') = ||x - x'||^2 = ||x - f_{\theta_d}(z)||^2 = ||x - f_{\theta_d}(f_{\theta_e}(x))||^2$. This loss is also called reconstruction error calculated by mean square error $MSE = \frac{1}{d} \sum_{i=1}^{d} (x_i - \hat{x}_i)^2$, where $d$ is the dimension of the input. In essence, this loss function aims to encode the input to a latent representation $z$ such that it can be regenerated by the decoder. To minimize the loss, common deep learning optimizers such as Adam [6] can be applied.

$$tr = \frac{MSE + \alpha}{\sqrt{\sigma(MSE)}}$$

**Anomaly Detection.** In the application of anomaly detection, we train a deep Autoencoder with benign IoT traffic data to learn IoT devices’ normal behavior, so that our Autoencoder could successfully extract and reconstruct features on benign samples but fails to do so on abnormal samples with unseen features. During detection phase, one input data sample that achieves a reconstruction error above a threshold will be detected as an abnormal data sample. In detail, after training Autoencoder on benign training dataset, we first calculate the reconstruction error ($MSE$) for each data sample from benign evaluation dataset, and then obtain the threshold by Equation 1, which computes the sum of the mean of $MSE$ plus standard deviation of $MSE$ over all evaluation samples (note that when calculating the reconstruction error with a mini-batch of $s$
samples, the standard deviation is divided by $\sqrt{s}$, and the scaled standard deviation is further multiplied by an coefficient $a$. The value of threshold should be as large as possible to suppress the majority of benign samples while preventing abnormal samples from being classified into benign samples. Extensive experiments show that the overall performance is the best when $a$ equals to 2.

2.4 FedDetect

We propose a federated learning algorithmic framework, FedDetect, for anomaly detection in distributed IoT devices. In this paper, we make the following assumptions: 1. The IoT device may be vulnerable but not initially be compromised. 2. 2 The Internet gateway (i.e. router) is not compromised. Distinctly from existing works on FL-based IoT anomaly detection, FedDetect utilizes adaptive optimizer (e.g., Adam) and cross-round learning rate scheduler, rather than naive FedAvg [10] for local training. Moreover, FedDetect supports both global threshold and personalized threshold for different scenarios. FedDetect is summarized as Algorithm 1.

Algorithm 1 FedDetect

1: Initialization $w_0$
2: for round $t = 0, 1, \ldots$ do
3: Adjust cross-round learning rate (cosine scheduler)
4: for client $i = 0$ to $K - 1$ do
5: $w_{t+1} = \text{Local Training with Adam}$
6: Upload $w_{t+1}$ to Server
7: end for
8: $w_{t+1} = \frac{1}{K} \sum_{i=0}^{K-1} w_{t+1}$
9: clients receive new model $w_{t+1}$
10: end for

11: Personalized Threshold Algorithm
12: for client $i = 0$ to $K - 1$ do
13: $tr^i = MSE^i + \frac{a}{\sqrt{s}} \sigma(MSE^i)$
14: end for

15: Globalized Threshold Algorithm
16: $MSE_{Global} = [MSE^0, \ldots, MSE^{K-1}]$
17: $tr_{Global} = MSE_{Global} + \frac{a}{\sqrt{s}} \sigma(MSE_{Global})$

Local Adaptivity and Cross-round Learning Rate Scheduler. The choice of Adam for local training and cross-round learning rate scheduler is based on our experimental observation. We empirically find that local Adam beats naive local SGD or SGD with momentum when applying a cross-round learning rate scheduler (e.g., cosine scheduler). The rationality of local adaptivity has also been intensively verified its efficacy in CV, and NLP tasks by recent theoretical works [14, 20].

Global and Personalized Threshold. After achieving the global model via federated training, we propose two algorithmic modules to calculate the anomaly threshold for each device: Global Threshold and Personalized Threshold. More specially, in the Global Threshold algorithm, each device runs the global model locally to get the $MSE$ sequence and then synchronizes it to the server, and the server uses $MSE$ sequences from all devices to generate a unified global threshold for detection in each device. For the Personalized Threshold algorithm, each device computes its local threshold using its local data only.

Global Threshold algorithm objectively expresses the detection performance of the FL-trained global model on the entire IoT networks (i.e., detecting a larger range of attack types as introduced in Section 2.2). Personalized Threshold algorithm can reflect the performance generalization of the global model on the local data of each device. Experimental results of these two algorithms demonstrate the efficacy of FL in diverse real-world scenarios (Section 3.4).

3 EXPERIMENTS

We evaluated FedIoT platform on two aspects: algorithmic performance in both global model and personalized model setting, a comprehensive analysis of the system efficiency, including computational speed, communication cost, and memory cost.

3.1 Setup

Implementation. We implemented two computing paradigms of FedIoT platforms: (1) IoT edge training, and (2) CPU/GPU-based distributed training. For the IoT edge training, we choose 9 Raspberry Pi 4B as client devices and a GPU server that integrates both the FL server and the MQTT service, as the design introduced in Section A. The Raspberry Pi 4B has a 1.5 GHz Broadcom BCM2711 processor and 4GB RAM. The GPU server has 2x AMD EPYC 7302 processors, an RTX A4000 GPU and 256 GB memory. The local training will be implemented on the Raspberry Pi and the weight integration will be implemented on the GPU server. For the CPU/GPU-based distributed computing, we used a GPU server that contains 4 NVIDIA RTX 2080Ti GPUs with sufficient GPU memory for our setting.

Dataset and Model Definitions. We evaluated FedIoT platform using two dataset described in Section 2.2. For the Autoencoder, we set the input with a dimension of 115, the same as the features of data. The encoder network has four hidden layers, as the dimension decreasing rate equals 75%, 50%, 33%, and 25%. The decoder has the same layer design as the encoder, but with an increasing sequence. The number of parameters for the Autoencoder is equal to 36628.

Hyper-parameters. We searched for the learning rate on a range of [0.1, 0.01, 0.001, 0.0001, 0.00001], input batch size on a range of [8, 16, 32, 64, 128], local training epoch on a range of [1, 5, 10, 15, 20], total training round on a range of [50, 100] and tested both $tanh$ and $sigmoid$ activation functions in Autoencoder. After hyper-parameter searching, we fixed the following hyper-parameters: the batch size for input is 64, the local training epoch in FL is 1, total training round is 100, and the activation function inside the Autoencoder is set as $tanh$ function. More hyper-parameters can be found in our source code.

3.2 Baselines

To evaluate our proposed algorithm in Section 2.4 comprehensively, we design the following three baselines:

• CL-Single: Each device trains a detection model with its own local dataset. This baseline can obtain the model performance
when training a local model without the help of federated learning.

- **CL-Multi**: A detection model is trained using the merged data of three devices, which have the top 3 performances when trained by CL-Single. This baseline is used to simulate the realistic scenarios that we can only centralize the data from a fraction of devices.
- **CL-Combined**: A detection model is trained with the merged data from nine devices. The result of this baseline serves as the upper bound performance for the centralized training. It may perform the best because it gathers all data samples from the entire IoT network.

### 3.3 Metrics

Following the existing works, we use three metrics to evaluate the detection performance: accuracy (ACC), precision (PR) and false positive rate (FPR). The formula and the confusion matrix will be shown in the Appendix.

### 3.4 Results of Learning Performance

**Evaluation using the global threshold.** We first evaluated the performance of FedDetect algorithm using the global threshold and compared its performance with three baselines. For the baseline CL-Single, we reported the average value of the nine devices’ model performances. The accuracy results shown in Figure 3a and 3b are evaluated on N-BaIoT dataset and LANDER dataset respectively. The full matrix evaluation plots and training curves are all listed in the Appendix. We could observe that as expected, the centralized training CL-Combined has the best performance in both dataset evaluations. It is clear that the FedDetect has a much better performance compared to the CL-Single and CL-Multi and achieves nearly upper bound performance compared to CL-Combined. In the evaluation on N-BaIoT dataset, the upper bound for the centralized training has accuracy of 94.89%, precision of 94.12%, and FPR of 5.26%. Meanwhile, the performance of the FedDetect has accuracy of 93.7%, precision of 88.2%, FPR of 11.9%. In the evaluation on LANDER dataset, the upper bound for the centralized training has accuracy of 96.76%, precision of 98.69%, FPR of 1.25%. On the other hand, the FedDetect achieves accuracy of 95.27%, precision of 93.81%, FPR of 6.39%. We could see that accuracy under the FL setting is nearly the same as upper bound performance in centralized training for both evaluations.

**Evaluation using the personalized threshold.** For FedDetect with personalized threshold, we evaluated its local performance on each edge device compared with the CL-Single baseline. As the results shown in Figure 3c, in the evaluation on N-BaIoT dataset, except for device A, FedDetect performs better or nearly equal to the CL-Single. The numbers listed above the bar are the relative difference between CL and FL settings. For example, device D and I achieve 0.126 and 0.281 increase of accuracy from FL, respectively. As the results shown in Figure 3d, in the evaluation on LANDER dataset, nearly all devices perform equally in both CL and FL settings. The number shows above the bar is the relative difference between the performance of FL and CL-Single. The detailed evaluation plots under personalized threshold are all shown in the Appendix.

![Figure 3: Experiment Results for Accuracy Performance over 4 experiment settings: (a)-(b) subfigures are evaluation under global threshold; (c)-(d) subfigures are evaluation under personalized threshold.](image)
convergence speed compared to CL, which could be seen from the training curve shown in the Appendix.

3.5 Analysis of System Efficiency
For the second part of the experiment, we evaluated the system performance of FedIoT with globalized threshold on the Raspberry Pi within N-BaIoT dataset.

Table 1: CPU/GPU Training v.s. IoT Edge Training

| Type               | Accuracy | Precision | FPR  |
|--------------------|----------|-----------|------|
| Simulation         | 0.937    | 0.882     | 0.119|
| Raspberry Pi       | 0.931    | 0.887     | 0.125|

We first verified that FedIoT on the real IoT device could achieve the same results as CPU/GPU distributed training. From Table 1, we could see that the results from the Raspberry Pi are nearly the same as the results from CPU/GPU simulation. The slight difference is due to different random initialization (i.e., different runs in different platforms).

We further tested the system efficiency on the Raspberry Pi platform. From Figure 4, we can see that the training time memory cost occupies only a small fraction of the entire host memory of Raspberry Pi (only 4G host memory). The training time per round is less than 1 minute. To understand the system cost more comprehensively, we analyzed the breakdown of the end-to-end training time shown in the Appendix.

Figure 4: Properties of Experiments on Raspberry Pi

We evaluate FedIoT platform with a simple but effective algorithm with both global threshold and personalized threshold for different scenarios. FedIoT supports both IoT edge training and CPU/GPU-based distributed training with the support of MQTT and MPI communication backend, respectively. We evaluate FedIoT platform and FedDetect algorithm with both global threshold and personalized threshold, and also analyze the system performance comprehensively. Our results demonstrate the efficacy of federated learning in detecting a large range of attack types, and the system efficiency analysis shows that both end-to-end training time and memory cost is affordable and promising for resource-constrained IoT devices.

4 RELATED WORKS
Our work is related to the application of federated learning in IoT cybersecurity. DiIoT [12] is the first system to employ a federated learning approach to anomaly-detection-based intrusion detection in IoT devices. IoTDefender [2] is another similar framework but obtains a personalized model by fine-tuning the global model trained with federated learning. [4] evaluates FL-based anomaly detection framework with learning tasks such as aggressive driving detection and human activity recognition. [8] further proposed an attention-based CNN-LSTM model to detect anomalies in an FL manner, and reduced the communication cost by using Top-k gradient compression. Recently, [15] even evaluates the impact of malicious clients under the setting of FL-based anomaly detection. Compared to these existing works, our FedIoT platform is the first work that analyzes both algorithmic and system performance in a real IoT platform.

5 CONCLUSION
In this paper, to further push forward the research in FL-based IoT cybersecurity, we build FedIoT platform with a simple but effective design philosophy. We apply Deep Autoencoder [17] as the model for anomaly detection to evaluate FL algorithms and our FedIoT platform. Moreover, we propose FedDetect, a federated learning algorithmic framework that utilizes adaptive optimizer and cross-round learning rate scheduler, rather than naive FedAvg [10] for local training. FedDetect supports both global threshold and personalized threshold for different scenarios. FedIoT supports both IoT edge training and CPU/GPU-based distributed training with the support of MQTT and MPI communication backend, respectively.

REFERENCES

[1] E. Bertino and N. Islam. 2017. Botnets and Internet of Things Security. Computer 50, 02 (feb 2017), 76–79. https://doi.org/10.1109/MC.2017.62
[2] Yuin Fan, Yang Li, Mengqi Zhan, Huaqin Cui, and Yan Zhang. 2020. IoTDefender: A Federated Transfer Learning Intrusion Detection Framework for 5G IoT. In 2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE). 88–95. https://doi.org/10.1109/BigDataSE50710.2020.00020
[3] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. 2008. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE’08). IEEE, 791–798.
[4] Rei Ito, Mineto Tsukada, and Hiroki Matsutani. 2020. An On-Device Federated Learning Approach for Cooperative Anomaly Detection.
[5] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019).
we could launch the FL on IoT platforms using a one-line API. They can exchange information with flexible APIs supported by FedIoT [20]. Jianyu Wang, Zheng Xu, Zachary Garrett, Zachary B. Charles, Luyang Liu, and Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. 2018. Communications-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA (Proceedings of Machine Learning Research, Vol. 54). Aarti Singh and Xiaojun (Jerry) Zhu (Eds.). PMLR, 1273–1282. http://proceedings.mlr.press/v54/mcmahan17a.html

[10] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Aracay. 2016. Federated Learning of Deep Networks using Model Averaging. CoRR abs/1602.05629 (2016). arXiv:1602.05629 http://arxiv.org/abs/1602.05629

[11] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbach, and Yuval Elowici. 2018. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. IEEE Pervasive Computing 17, 3 (2018), 12–22. https://doi.org/10.1109/MPRV.2018.3036773

[12] Thien Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N. Asokan, and Ahmad-Reza Sadeghi. 2019. IoDT: A Federated Self-Learning Anomaly Detection System for IoT. 756–767. https://doi.org/10.1109/ICDCS.2019.00080

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703 (2019).

[14] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H Brendan McMahan. 2020. Adaptive federated optimization. arXiv preprint arXiv:2003.00285 (2020).

[15] Valerian Rey, Pedro Miguel Sánchez Sánchez, Albert Huertas Celdrán, Gérome Bovet, and Martin Jaggi. 2021. Federated Learning for Malware Detection in IoT Devices. arXiv preprint arXiv:2104.09994 (2021).

[16] Khaleed Riad, Teng Huang, and Lishan Ke. 2020. A dynamic and hierarchical access control for IoT in multi-authority cloud storage. Journal of Network and Computer Applications 160 (04 2020), 102633. https://doi.org/10.1016/j.jnca.2020.102633

[17] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.

[18] Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. 2018. Communication compression for decentralized training. arXiv preprint arXiv:1803.06443 (2018).

[19] USC LANDER. 2020. 10-day Operational IoT Traces. http://www.isi.edu/ant/lander

[20] Jianyu Wang, Zheng Xu, Zachary Garrett, Zachary B. Charles, Luyang Liu, and Xiaojin (Jerry) Zhu (Eds.). PMLR, 1273–1282. http://proceedings.mlr.press/v54/mcmahan17a.html

The FL client (IoT device) is maintained as an instance by the FL aggregation functions, and the FL server is managed by the MQTT server. As such, we deploy two servers: the MQTT server for message passing and the FL server for federated aggregation. The model weight exchange is by MQTT messaging server, while control messages such as device registration and dataset distribution are transmitted through the FL server. FedIoT platform also supports CPU/GPU-based distributed training. In this computing paradigm, we can evaluate the algorithm prototype before deploying the algorithm to IoT devices. CPU/GPU-based distributed training is supported by the MPI communication backend.

In our experiments, we have analyzed the system performance and also verified that our platform can provide consistent results in two environments by comparing the results from the CPU/GPU-based distributed training and IoT edge training (Section 3.5).

B DATA DISTRIBUTION

In our paper, we introduced the UCI N-BaIoT dataset [11] and USC LANDER IoT Operation Traces-20200127 dataset [19] for the system evaluation.

N-BaIoT dataset [11] is a widely used public dataset for research on anomaly detection of IoT data. N-BaIoT captures network traffic flow from 9 commercial IoT devices (Table 6) authentically attacked by network intrusions. In the original N-BaIoT dataset, it provides 115 statistic features, which could be severely influenced by the hostile attack, as shown in Table 3. Each IoT device has 2 subsets: one benign set containing normal network flow data only, and one attack data subset consisting of two common malware attacks, Mirai and BASHLITE, which each contain five different kinds of attack types (Table 4).

USC LANDER IoT Operation Traces-20200127 dataset [19] is one of the latest datasets for the research of the operational traffic on IoT edge devices. The LANDER dataset contains 10-day operational traffic for 14 different widely-used IoT devices (Table 7) located in a LAN network without any types of attack. All IoT devices are initially off and get boot up right after measurement starts. Traces are captured at LAN port of the LAN router. All non-IoT traffic are removed for privacy concern. In the LANDER dataset, it provides the original traffic capture files (i.e. pcap file) for each device, which could not be directly fed into the model. For the sake of consistency of the experiments, we extract the same features as N-BaIoT dataset does, as shown in Table 3. Due to the massive volume of the original dataset, we only intercept the first 10% of the underlying abstract communication layer. By following this client-oriented programming design pattern, users can define their own client/server behavior when participating in training or coordination in the FL algorithm, so that FedIoT allows users to focus on algorithmic implementations instead of considering the low-level communication mechanism. In the infrastructure layer, FedIoT supports AI-enabled IoT edge devices and the lightweight communication APIs for distributed system. With such testbeds built upon real-world hardware platforms, researchers can evaluate realistic system performance, such as training time, communication, and computation cost. Moreover, for FedIoT, researchers only need to program with Python to customize their research experiments without the need to learn new system frameworks or programming languages (e.g., Java, C/C++). MQTT [3] is supported for IoT setting. As such, we deploy two servers: the MQTT server for message passing and the FL server for federated aggregation. The model weight exchange is by MQTT messaging server, while control messages such as device registration and dataset distribution are transmitted through the FL server.

FedIoT platform also supports CPU/GPU-based distributed training. In this computing paradigm, we can evaluate the algorithm prototype before deploying the algorithm to IoT devices. CPU/GPU-based distributed training is supported by the MPI communication backend.
Table 3: Description of Data Features

| Packet Properties | Statistic                           | Aggregated By         | Features Number |
|-------------------|-------------------------------------|-----------------------|-----------------|
| size of outbound packets | mean, Var                           | IP, MAC-IP, channel, Socket | 40              |
| number             | number count                        | IP, MAC-IP, channel, Socket | 20              |
| time latency       | mean, Var, elapsed time             | channel               | 15              |
| size of inbound and outbound packets | magnitude, radius, covariance, correlation coefficient | channel, socket | 40              |

data for our experiment. Based on the MAC and HOST address information in LANDER, we generate some malicious attack data by the source code of the Mirai UDP attack which targets to the listed devices in the dataset.

C  SYSTEM PERFORMANCE

In this paper, we evaluated FedIoT platform on two aspects: algorithmic performance in both global model and personalized model setting; a comprehensive analysis of the system efficiency, including computational speed, communication cost, and memory cost. We implemented two computing paradigms of FedIoT platforms: (1) IoT edge training as shown in Figure 5, and (2) CPU/GPU-based distributed training.

In the evaluation part, following the existing works, we use three metrics to evaluate the detection performance: accuracy (ACC), precision (PR) and false positive rate (FPR). The formula is shown in Equation 2, Equation 3 and Equation 4, which is explained in the confusion matrix in Table 5.

\[ ACC = \frac{TP + TN}{TP + TN + FP + FN} \]  
\[ PR = \frac{TP}{TP + FP} \]  
\[ FPR = \frac{FP}{FP + TN} \]

For the evaluation on learning performance, we performed the experiments on both the global threshold setting and personal threshold setting. The results shown in Figure 6 and Figure 7 are evaluated on N-BaIoT dataset and LANDER dataset respectively for global threshold. The subfigures (d)-(f) in both figures show the training curves over 100 epochs (FL uses rounds) in both CL and FL settings. Figure 8 and Figure 9 are evaluated on N-BaIoT dataset and LANDER dataset respectively for personalized threshold. The numbers listed above the bar are the relative difference between CL and FL settings.

Table 4: Attack Types in Malicious Dataset

| Attack | Mechanism                                                   |
|--------|-------------------------------------------------------------|
| SCAN   | scanning the network for vulnerable devices                 |
| JUNK   | sending junk data                                          |
| COMBO  | sending junk data and opening a connection to a specified IP address and port |
| TCP    | TCP flooding                                                |
| ACK    | ACK flooding                                                |
| SYN    | SYN flooding                                                |
| UDP    | UDP flooding                                                |
| UDPPLAIN | UDP flooding with fewer options, optimized for higher PPS    |

Table 5: The Confusion Matrix for Two Classes

| Predicted Benign | Predicted Malicious |
|------------------|--------------------|
| Actual Benign    | True Negative (TN) | False Positive (FP) |
| Actual Malicious | False Negative (FN)| True Positive (TP)  |

\[ ACC = \frac{TP + TN}{TP + TN + FP + FN} \]  
\[ PR = \frac{TP}{TP + FP} \]  
\[ FPR = \frac{FP}{FP + TN} \]  

Figure 5: Hardware platform
### Table 6: Overview of IoT Devices and Data Distribution in N-BaIoT Dataset

| Device ID | Device Make | Device Type | Total Number | Training Set Number | Opt Set Number | Benign Number | Attack Number |
|-----------|-------------|-------------|--------------|---------------------|----------------|---------------|---------------|
| A         | Danmini     | Doorbell    | 49548        | 33197              | 16351          | 5000          | 5000          |
| B         | Ennio       | Doorbell    | 13113        | 8786               | 4327           | 5000          | 2500          |
| C         | Ecobee      | Thermostat  | 39100        | 26197              | 12903          | 5000          | 5000          |
| D         | Philips B120N/10 | Baby monitor | 175240      | 117411            | 57829          | 5000          | 5000          |
| E         | Provision PT-737E | Camera    | 62154        | 41643             | 20511          | 5000          | 5000          |
| F         | Provision PT-838 | Camera    | 98514        | 66004             | 32510          | 5000          | 5000          |
| G         | Samsung SNH 1011 N | Webcam | 52150        | 34940             | 17210          | 5000          | 2500          |
| H         | SimpleHome XCS7-1002 | Camera | 46585        | 31212             | 15373          | 5000          | 5000          |
| I         | SimpleHome XCS7-1003 | Camera | 19528        | 13084             | 6444           | 5000          | 5000          |

### Table 7: Overview of IoT Devices and Data Distribution in LANDER Dataset

| Device ID | Device Make | Device Type | Total Number | Training Set Number | Opt Set Number | Benign Number | Attack Number |
|-----------|-------------|-------------|--------------|---------------------|----------------|---------------|---------------|
| A         | Amcrest Cam | Camera      | 181836       | 121830             | 60006          | 5000          | 5000          |
| B         | Belkin Plug | Plug        | 14969        | 10029              | 4940           | 4940          | 4940          |
| C         | D-Link Cam  | Camera      | 122677       | 82194              | 40483          | 5000          | 5000          |
| D         | Dyson Purifier | Air purifier | 2157        | 1445               | 712            | 712           | 712           |
| E         | Foscam Cam  | Camera      | 14563        | 9757               | 4806           | 4806          | 4806          |
| F         | Foscam Cam2 | Camera      | 15778        | 10571              | 5207           | 5000          | 5000          |
| G         | HP Printer  | Printer     | 11709        | 7845               | 3864           | 3864          | 3864          |
| H         | Philips Bulb | Bulb        | 85967        | 57598              | 28369          | 5000          | 5000          |
| I         | Samsung Cam | Camera      | 36640        | 24549              | 12091          | 5000          | 5000          |
| J         | Tenvis Cam  | Camera      | 24077        | 16132              | 7945           | 5000          | 5000          |
| K         | TPlink Bulb | Bulb        | 4964         | 3326               | 1638           | 1638          | 1638          |
| L         | TPlink Plug | Plug        | 6368         | 4267               | 2101           | 2101          | 2101          |
| M         | Wansview Cam | Camera | 173764       | 116422             | 57342          | 5000          | 5000          |
| N         | Wyze Cam    | Camera      | 87026        | 58307              | 28719          | 5000          | 5000          |
Figure 6: N-BaIoT Dataset Experiment Results: (a)-(c) subfigures are evaluation over 4 experiment settings; (d)-(f) subfigures shows results over 100 epochs (FL uses rounds).

Figure 7: LANDER Dataset Experiment Results: (a)-(c) subfigures are evaluation over 4 experiment settings; (d)-(f) subfigures shows results over 100 epochs (FL uses rounds).
Figure 8: Results of Personalized Evaluation on N-BaIoT dataset

Figure 9: Results of Personalized Evaluation on LANDER dataset