Secure Federated Training: Detecting Compromised Nodes and Identifying the Type of Attacks

Pretom Roy Ovi[a], Aryya Gangopadhyay[b], Robert F. Erbacher[c], Carl Busart[d]
Center for Real-time Distributed Sensing and Autonomy
[a, b] University of Maryland, Baltimore County, USA
[c, d] U.S. DEVCOM Army Research Laboratory
{[a]povi1, [b]gangopad}@umbc.edu, {[c]Robert.F.Erbacher.civ, [d]carl.e.busart.civ}@army.mil

***Abstract*—Federated learning (FL) allows a set of clients to collaboratively train a model without sharing private data. As a result, FL has limited control over the local data and corresponding training process. Therefore, it is susceptible to poisoning attacks in which malicious clients use malicious training data or local updates to poison the global model. In this work, we first studied the data level and model level poisoning attacks. We simulated model poisoning attacks by tampering the local model updates during each round of communication and data poisoning attacks by training a few clients on malicious data. And clients under such attacks carry faulty information to the server, poison the global model, and restrict it from convergence. Therefore, detecting clients under attacks as well as identifying the type of attacks are required to recover the clients from their malicious status. To address these issues, we proposed a way under federated framework that enables the detection of malicious clients and attack types while ensuring data privacy. Our clustering-based approach utilizes the neuron's activations from the local models to identify the type of poisoning attacks. We also proposed to check the weight distribution of local model updates among the participating clients to detect malicious clients. Our experimental results validated the robustness of the proposed framework against the attacks mentioned above by successfully detecting the compromised clients and the attack types. Moreover, the global model trained on MNIST data couldn't reach the optimal point even after 75 rounds because of malicious clients, whereas the proposed approach by detecting the malicious clients ensured convergence within only 30 rounds and 40 rounds in independent and identically distributed (IID) and non- independent and identically distributed (non-IID) setup respectively.**

***Index Terms*—data level poisoning, model level poisoning, federated training, adversarial attacks, identify attacks**

I. Introduction

Distributed machine learning has received much attention [1], [2] over the years. In [3], researchers examined distributed machine learning algorithms for multiple data centers located in various areas. But not much thought has been given to how secure the data is and the impact of data distribution on the performance. So, maintaining data security has always been a critical research problem. Research in this area has been accelerated significantly with the advent of federated learning. Nowadays, FL is being applied in a large number of fields [4]–[7]. The main idea behind this training is that, instead of uploading and storing the entire set of data, which may invoke security issues, there will be a client-server based model [8]–[10]. This approach makes it possible to extract knowledge from the data distributed on local devices. In recent times, the standard federated learning system has been expanded in several ways such as FedAvg [8], SMC- AVG [11], FedProx [12].

Even though FL takes steps to address privacy concerns when training models with real-world data, there are still concerns over how robust the models are. In the past few years, there has been a lot of research demonstrating the vulnerability of deep neural networks to attacks. Due to its distributed nature, FL is fundamentally vulnerable to model poisoning attacks [13]–[16]. In model poisoning attack, it is assumed that an authentic client will compute local model updates based on genuine training data and further manipulate the local model updates before sending them to the central server to poison the model. On the other hand, data poisoning attacks assume that malicious clients will be trained on dirty labeled data and local model updates based on wrongly labeled data restrict the global model from converging. In such a scenario, it is challenging for the server to thoroughly examine the data used for model training. Authors in [17] pointed out that dirty-label data poisoning attacks tend to cause a lot of misclassifications, up to 90%, when an attacker adds a small number of dirty-label samples to the training dataset. And authors in [18] pointed out the data poisoning attacks in FL as an issue that needs immediate addressing.

Existing literature showed the vulnerability of deep model in FL framework against the aforementioned attacks, and there lies a research gap in identifying the type of attack. Based on the real world use cases and application scenarios, FL is of two kinds, namely cross device FL and cross silo FL. In the cross device FL setup (designed for mobile devices), there could be thousands of clients and we can easily discard the malicious clients during aggregation. However, in the cross silo federated setup (designed for organizations), a small number of malicious clients can easily poison the global model because of very limited number of total organizations/clients. Clients are very limited in this setup, and so detecting the clients under attacks is not enough. Rather, we should identify the type of attacks and based on the attack types, further action can be taken to recover the clients from malicious activities. So, besides detecting the malicious clients, identifying the attack types is also important specially in cross-silo FL setup.

Our work addresses this problem by identifying the malicious clients and the attack type. The major contributions we have included in this work are:

- *Effect of Compromised Clients on Convergence:* In the cross-silo federated setup with a small number of clients, first we showed how compromised clients affect the convergence of the global model. Experimental results suggested that even a small number of malicious clients can have a significant negative effect on the global model.
- *Detection of Compromised Clients under Attacks:* Secondly, we proposed an approach to detect malicious clients carrying faulty information to the server during model training. And by comparing the weight distribution of local model updates among the clients, the proposed framework successfully detected all the compromised clients.
- *Detection of the type of Attacks:* Besides detecting malicious clients, our proposed clustering based approach enables determining whether a client has been poisoned at the data level or the model level while ensuring data privacy. Label-flipping and dirty-labeling are two ways to launch data poisoning attacks, and our proposed method can identify both types of data poisoning attacks.
- *Performance Analysis on IID and non-IID manner:* After detecting the compromised clients and discarding them during aggregation, our approach propelled a global model accuracy of approximately 98.26% and 96.05% respectively while classifying MNIST digit classes in both IID and non-IID setup.

II. Related Work

In this section, we will go over the recent advancements of federated learning algorithms, and a summary of previous works on the adversarial attacks on federated learning.

Deep learning models, by their very nature, necessitate a substantial amount of data to make acceptable predictions. This specific requirement gives rise to a potential problem of data breach. Federated learning was first introduced in [9] as a potential solution to this problem [12]. The architecture of vanilla federated learning was first outlined in [19]. Because of the internal architecture of federated framework, FL allows deep model to train on the client side without uploading private data to the clouds. Since then, a substantial amount of research has been conducted on the building blocks i.e., client selection [20], communication between client and server [21], aggregation schemes [22]. The selection of clients is an essential component of a federated learning environment. The client represents the devices or organizations that will help to build the global model with their trained weights. As a result, a secure and efficient communication structure is required to complete the weight transfer between server and client. Finally, a suitable update aggregation method is also required to incorporate client-side knowledge into the server properly. Two of the common schemes are synchronous [8] and asynchronous [18] aggregation, and either of those can be chosen based on the application scenario.

FL is vulnerable to data poisoning and model poisoning attacks. Label-flipping [23] and backdoor attacks [13], [17] are examples of data poisoning attacks. Attacker can easily manipulate its local data by directly swapping the labels of honest training instances of one class (the source class) to a specific target class while keeping the features of the training data unchanged. It is known as the label-flipping attack [24], [25], which can cause significant drops in the performance of global model even with a small percentage of malicious clients [25]. A backdoor poisoning attack was proposed in [13], [26], in which the attacker injects backdoored inputs into local data to modify specific features of training data and implant backdoors in the global model. An aggregation-agnostic attack, which continuously adds noises to the model parameters to introduce backdoors and restrict the global model from converging, was developed in [27]. But identifying the type of poisoning attacks from the server end in FL framework is not studied yet. So, the research gap lies in this direction.

III. Methodology

In this section, we describe the detailed architecture and work flow of our proposed federated training set up.

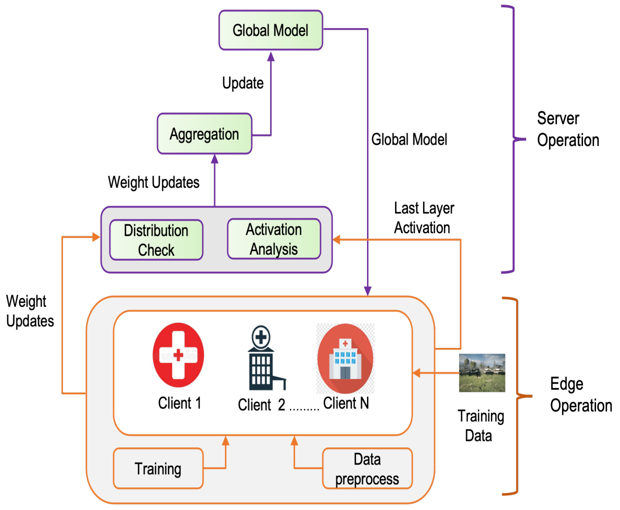

Fig. 1: Proposed Federated Framework

Our proposed framework is built on top of the FedAvg [8] algorithm where some of the notations used in this description are $\mathcal{N} = \{1, \ldots, N\}$ signify the set of N clients, each of which has their own dataset $D_{k\in\mathcal{N}}$. Each of them trains a local model using their own dataset and only shares the model parameters with the FL server. Then, the global model, $\mathbf{w}_G$ formation takes place with all the local model updates which can denoted by $\mathbf{w} = \cup_{k\in\mathcal{N}} \mathbf{w}_k$. The proposed federated framework is depicted in figure 1 and complete pseudocode of our method is shown in algorithm 1. The process based on the workload of server and client is described below:

1) **Executed at the client level**
    - *Local training and updated parameter transmission:* Each client will be trained on local training samples

to learn parameters after receiving the global model $\mathbf{w}_G^t$ from the server, where t stands for each iteration index. For each client image data D^i are fed to the local model for training. The client tries to minimize the loss function [18] $L\left(\mathbf{w}_k^t\right)$ and searches for optimal parameters $\mathbf{w}_k^t$.

$$\mathbf{w}_k^{t^*}=\underset{\mathbf{w}_k^t}{\arg\min} L\left(\mathbf{w}_k^t\right) \tag{1}$$

After each round of training, updated local model parameters are sent to the server afterwards. In addition, each client will sent the last "Relu" layer activation value of local training samples that will be utilized to detect data level poisoning.

2) **Executed in server**

- *Weight initialization:* The server determines the type of application and how the user will be trained. Based on the application, the global model is built in the server. The server then distributes the global model $\mathbf{w}_G^0$ to selected clients.
- *Weight distribution:* According to the proposed framework, updated local parameters received by the server from the clients will be used to detect the anomaly before aggregation. For each participant, we'll look at the weight distribution of local model parameters and discard those clients whose distributions deviate significantly from the distribution of majority clients.
- *Activation analysis:* In this block, last layer's (prior to softmax layer) activation of all local models will be utilized by a cluster based approach to detect the data level poisoning attack (Details in section IV(C)).
- *Aggregation and global update:* The server aggregates the local models from the participants by discarding the detected compromised clients and then sends the updated global model parameters $\mathbf{w}_G^{t+1}$ back to the clients. The server wants to minimize the global loss function [18] $L\left(\mathbf{w}_G^t\right)$, i.e.

$$L\left(\mathbf{w}_G^t\right)=\frac{1}{N}\sum_{k=1}^{N} L\left(\mathbf{w}_k^t\right) \tag{2}$$

This process is repeated until the global loss function converges or a desirable training accuracy is achieved. The Global Updater function runs on the SGD [28] formula for weight update. The formal equation of global loss minimization formula by the averaging aggregation at the t^{th} iteration is given below:

$$\mathbf{w}_G^t=\frac{1}{\sum_{k\in\mathcal{N}} D_k}\sum_{k=1}^{N} D_k\mathbf{w}_i^t \tag{3}$$

Algorithm 1: Algorithm for proposed federated training

```
Require: Clients number $K$ per iteration, local epochs
         number $E$, and learning rate $\mu$, local
         minibatch size $B$, image data $D^i$
Ensure: Global model $\mathbf{w}_G$
1  [Server]
2  Initialize $\mathbf{w}_G^0$
3  Global Updater ():
4  for each iteration t from 1 to $T$ do
5      Randomly choose a subset $\mathcal{S}_t$ of $K$ clients from $\mathcal{N}$
6      for each client $k \in \mathcal{S}_t$ parallely do
7          $\mathbf{w}_k^{t+1} \leftarrow$ LocalTraining $(k, \mathbf{w}_G^t)$
8          $\mathbf{a}_k^{t+1} \leftarrow$ LocalTraining $(k, \mathbf{w}_G^t)$
9      end
10     check weight distribution $\mathbf{w}_K^{t+1}$ (Detecting
         compromised nodes)
11     $\mathbf{w}_G^t = \frac{1}{\sum_{k\in\mathcal{N}} D_k}\sum_{k=1}^{N} D_k \mathbf{w}_k^t$ (Aggregation through
         average after discarding compromised nodes)
12 end
13 [Client $k$]
14 LocalTraining$(k, \mathbf{w})$ :
15 Split local datasets $D_k^i$ where $D_k^i \in D_k$ to minibatches
     of size $B$ which are included into the set $\mathcal{B}_k$, local
     activation $\mathbf{a}_k$
16 for each local epoch j from 1 to $E$ do
17     for each $b \in \mathcal{B}_k$ do
18         $\mathbf{w} \leftarrow \mathbf{w} - \mu\Delta G(\mathbf{w}; b)$ ($\Delta G$ is the gradient on
             $b$.)
19         $\mathbf{a} \leftarrow$Act$(\mathbf{wx}+\mathbf{b})$ (Last layer Activation prior
             to softmax layer.)
20     end
21 end
22 send $\mathbf{w}$ to server
23 send $\mathbf{a}$ to server
```

IV. Experimental Setup and Result Analysis

This section will go through the procedures to set up the experiments and the respective result analysis.

In our experiment, federated learning based deep model is implemented on Keras running on TensorFlow backend. We built a federated learning framework to train the model with no shared data with the server. To initiate the training, the server sends a global model to all the participating clients, sets the training pace and how the users will be trained, and then aggregates the updated weights from the participating clients to update the global model. The early stopping criteria controls the total number of rounds. And for each round, 1 epoch of local training is conducted on client side. We chose image classification as application and experimented with VGG-16 deep learning model on MNIST [29] digit dataset, which includes 10 digit classes from 0 to 9. We simulated FL training with only 15 clients to conduct the experiment.

A. Effect of Compromised Clients on Convergence

In our experiment, we simulated the FL training environment in such a way that among the total 15 clients, 12 clients were authentic and rest of the 3 clients were compromised. And

among the 3 compromised clients, 1 client was the victim of model poisoning attack, and 2 clients were attacked with data poisoning attacks.
To simulate the model poisoning attack on one client, we tampered the real weights of that client by adding random values at the time of weight sharing with the server. And we simulated data level poisoning attack on 2 clients by label flipping for one client and dirty labeling for another. Data level poisoning also induces model poisoning. By sharing the faulty weights with the server, these 3 compromised clients restrict the global model from converging.

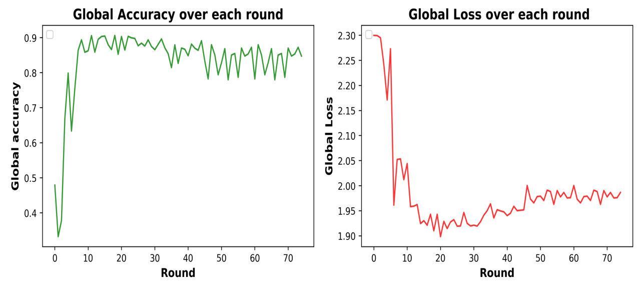

Fig. 2: Effect of Compromised nodes on Convergence, Global Accuracy (left) and Global Loss (right) over Rounds.

And to demonstrate the effect of compromised nodes in the cross-silo federated setup, we depicted the experimental results in figure 2. In figure 2, the global model couldn't reach the optimal point even after 75 rounds with only 3 malicious clients out of 15 clients, whereas after discarding the malicious clients, it reached to global optima with only 30 and 40 rounds in IID and non-IID setup respectively, depicted the result in figure 3.

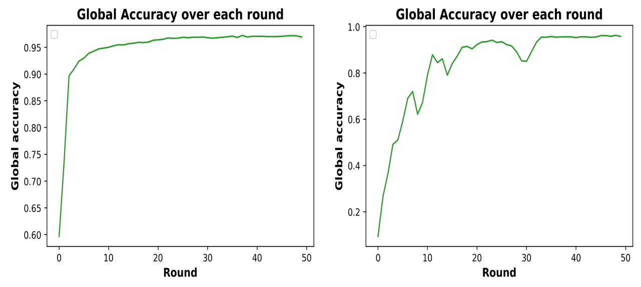

Fig. 3: Global Accuracy over rounds on the test set in IID (Left), and non-IID setup (Right).

B. Detection of Compromised Nodes under Attacks

In targeted poisoning attacks, a common observation is that the model updates from malicious clients have unique characteristics compared to the ones from honest clients. We have included two analyses to demonstrate that our proposed federated framework can detect the compromised nodes.

The weight vectors, shared with the server every round, are very high dimensional, and so, using dimensions reduction

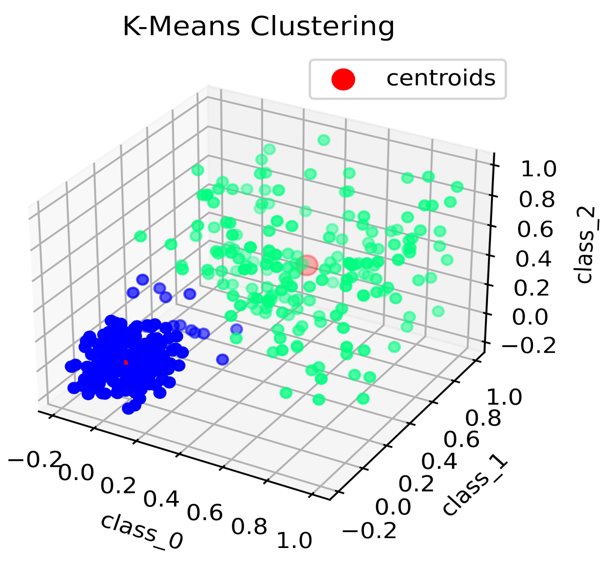

Fig. 4: K-means clustering: blue and green cluster for the data points from honest and compromised clients respectively

techniques such as principal component analysis (PCA), we plotted the last layer weight vector of all clients to make 2 clusters in the k-means clustering, depicted in figure 4. And we got a green cluster of weights (for compromised clients) and a blue cluster of weights (for honest clients) based on two different weight distribution. But from such clustering, we get the assumptions of number of data points which is faulty and/or authentic. But k-means clustering can not detect the compromised clients because there is no way to look at the weight distribution for individual clients. Moreover, cluster-based defensive strategies cannot differentiate between model updates from malicious clients and honest clients in non-IID setup.

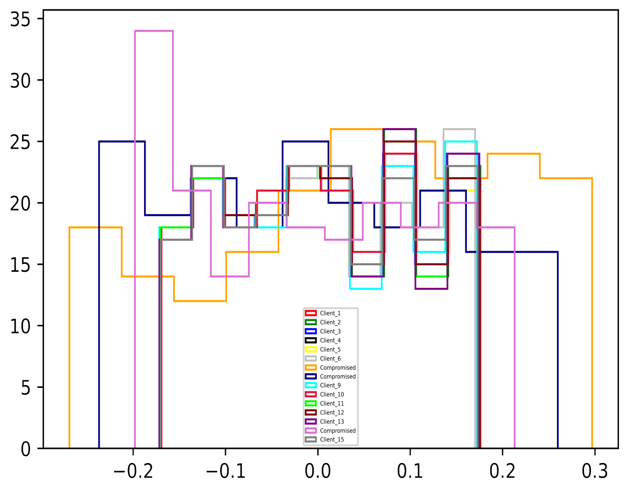

Fig. 5: Weight distribution of participating clients.

So, to detect the compromised client, we need to look into the weight distribution of each client. Assigning each client with unique color, we plotted the weight distribution for all 15 clients, showed in figure 5. From this figure, it is clearly observed that among all 15 clients, 3 clients (assigned with orange, blue, pink colors and named as "compromised" client) have a somewhat different distribution than majority clients. And we detected all those 3 malicious clients under attacks from the weight distribution. But still, we can not say which clients were affected by model poisoning and which were affected by data poisoning.

C. Identify the Type of Attacks

To detect the type of attack, besides weight sharing, each client will be asked to share the activations of last layer neurons prior to the softmax layer with the server because the last layers of any deep model carry out comparatively more refined and less interpretable information than initial layers. Since the neurons' activation of deep model can be high dimensional, each client will perform dimension reduction technique e.g. Principal Component Analysis (PCA), on its activation value to convert it to lower dimension and share with the server. And sharing only the last layers' activation even after converting to lower dimensions will not violate the data privacy of federated learning protocol. So, here the server utilized the neurons' activation shared by the clients to identify if any particular client is being trained on wrongly labeled data to launch data level poisoning.

In the above subsection, we detected 3 compromised client but were not able to identify the type of attacks. As stated earlier, we simulated data level poisoning attack on 2 clients using label flipping for one client and dirty labeling for another. Since we used MNIST dataset for our experiment, all 15 clients should be trained on MNIST data. But, to launch the data poisoning attack, we intentionally trained one client (client_3) on MNIST but flipped the label of digit 0 and digit 1 class. And we trained other client (client_4) on dirty labeled data where images of public transport e.g. bus, metro and battlefield vehicles e.g. helicopter, tank have been labeled as digit classes. Here, we employed a cluster based representation for each client and compared the class-wise cluster among the clients to detect the data level poisoning. Training samples from the same class label is also expected to have the same space in cluster based representation. And for simplicity and easy visualization, we visualized only five digit classes instead of 10 digit classes. And figure 6 illustrates that client_1 and client_2 are honest and trained on their local MNIST digit data, and it is clearly visible that class-wise cluster position in 3D space is almost identical for both of them. In figure 6 (lower left), for client_3, the blue and orange clusters (assigned for digit 0 and digit 1 class respectively) have swapped their position compared with the honest clients because label flipping attack was conducted on digit 0 and digit 1 classes for this particular client. But clusters for other classes were almost in the same space for client_3 in comparison with client_1 and client_2 because of true labeling for other digit classes. And for client_4, depicted

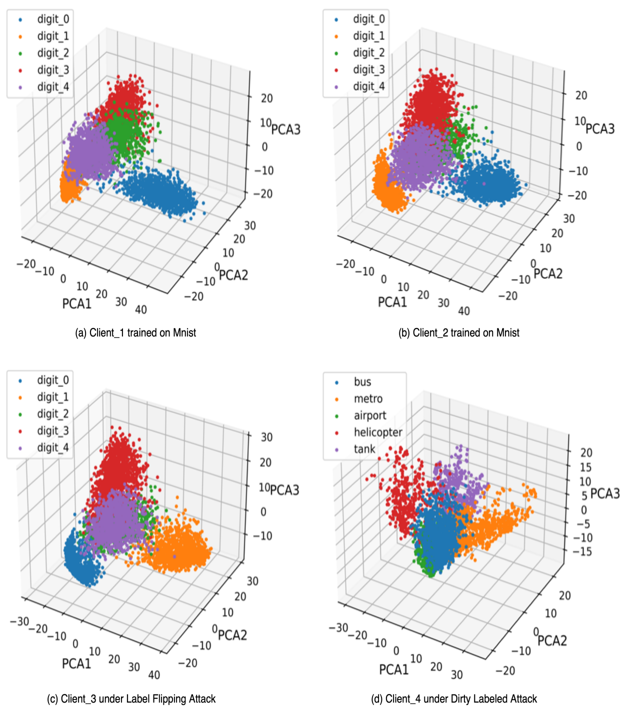

Fig. 6: Clustering based approach: (a) honest client, (b) honest client, (c) client attacked with label flipping between digit-0 and digit-1 class, (d) client attacked with dirty labeling.

in figure 6 (lower right), class-wise clustering positions for all classes were different compared to the honest clients because client_4 was trained on dirty labeled data where the images of bus, metro, airport, helicopter, tank were labeled as digit classes. So from the detected 3 compromised clients (in section IV B), our cluster based approach successfully identified data level poisoning attacks on two clients. And rest of the compromised clients is under model poisoning attack.

D. Performance Analysis on IID and non-IID Data

After detecting the malicious clients and discarding them into model aggregation, we evaluated the efficacy of our FL framework on two different data scenarios, i.e., independent and identically distributed (IID) data and non-independent and identically distributed data (non-IID). The fundamental difference between the two scenarios lies in each random variable's distribution and mutual independence. It is highly unlikely to achieve a situation where equal data distribution among all clients is guaranteed in a real environment. It is even possible that the imbalanced data distribution might not contain all the classes throughout the clients. In IID setup, all clients were given the training samples with a comparatively balanced class distribution. And to design the non-IID experimentation, we simulated an extreme non-IID setup by assigning only two classes per client. It is thus evident that in non-IID setup, the neural network should take considerably more rounds to reach the same performance level as the IID scenario. As illustrated in figure 3, our analysis further adds credence to this hypothesis. For example, in the non-IID scenario, we reached the saturation point somewhere after 40 rounds, whereas the IID scenario requires 25% fewer rounds (only 30 rounds) to achieve the saturation level. Additionally, non-IID setup (96.05% accuracy) has a slight (around 2%) decrease in performance compared to IID setup (98.12%) due to the class imbalance among the clients impacting the global network.

V. Conclusion

In this work, to reveal the vulnerability of the federated training against model poisoning and data poisoning attacks, we first showcased the effect of malicious clients on the global model's convergence. Secondly, we detected the compromised clients from the server end by comparing the weight distribution of local model updates. Thirdly, the proposed clustering-based approach identified the data level poisoning attack from the last layer neurons' activation while ensuring data privacy. Finally, our proposed framework has been trained on MNIST data in both IID and non-IID data fashion after discarding the detected compromised clients during aggregation.

ACKNOWLEDGMENT

This research is partially supported by NSF Grant No. 1923982 and U.S. Army Grant No. W911NF21-20076.

References

[1] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang *et al.*, "Large scale distributed deep networks," *Advances in neural information processing systems*, vol. 25, 2012.

[2] M. Duan, K. Li, X. Liao, and K. Li, "A parallel multiclassification algorithm for big data using an extreme learning machine," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2337–2351, 2017.

[3] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia:{Geo-Distributed} machine learning approaching {LAN} speeds," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 629–647.

[4] P. R. Ovi, E. Dey, N. Roy, A. Gangopadhyay, and R. F. Erbacher, "Towards developing a data security aware federated training framework in multi-modal contested environments," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications IV*, vol. 12113. SPIE, 2022, pp. 189–198.

[5] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen *et al.*, "Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data," *Scientific reports*, vol. 10, no. 1, pp. 1–12, 2020.

[6] H.-A. Rashid, P. R. Ovi, A. Gangopadhyay, and T. Mohsenin, "Tinym2net: A flexible system algorithm co-designed multimodal learning framework for tiny devices," *ArXiv*, 2022.

[7] H. Cho, A. Mathur, and F. Kawsar, "Flame: Federated learning across multi-device environments," *arXiv preprint arXiv:2202.08922*, 2022.

[8] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[9] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[10] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[11] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *arXiv preprint arXiv:1611.04482*, 2016.

[12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[13] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[14] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[15] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[16] C. Xie, O. Koyejo, and I. Gupta, "Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 261–270.

[17] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[18] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[19] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[20] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[21] P. P. Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency, "Think locally, act globally: Federated learning with local and global representations," *arXiv preprint arXiv:2001.01523*, 2020.

[22] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data *et al.*, "A field guide to federated optimization," *arXiv preprint arXiv:2107.06917*, 2021.

[23] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[24] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 301–316.

[25] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *European Symposium on Research in Computer Security*. Springer, 2020, pp. 480–501.

[26] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[27] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[28] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 8–16, 2020.

[29] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.