2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)

Understanding Distributed Poisoning Attack in Federated Learning

Di Cao*, Shan Chang*, Zhijian Lin*, Guohua Liu*, Donghong Sun†
*Computer Science and Technology, Donghua University, Shanghai, China
†Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China
caodi@mail.dhu.edu.cn, changshan@dhu.edu.cn, lzj@mail.dhu.edu.cn, ghliu@dhu.edu.cn, sundonghong@tsinghua.edu.cn

***Abstract*—Federated learning is inherently vulnerable to poisoning attacks, since no training samples will be released to and checked by trustworthy authority. Poisoning attacks are widely investigated in centralized learning paradigm, however distributed poisoning attacks, in which more than one attacker colludes with each other, and injects malicious training samples into local models of their own, may result in a greater catastrophe in federated learning intuitively. In this paper, through real implementation of a federated learning system and distributed poisoning attacks, we obtain several observations about the relations between the number of poisoned training samples, attackers, and attack success rate. Moreover, we propose a scheme, Sniper, to eliminate poisoned local models from malicious participants during training. Sniper identifies benign local models by solving a maximum clique problem, and suspected (poisoned) local models will be ignored during global model updating. Experimental results demonstrate the efficacy of Sniper. The attack success rates are reduced to around 2% even a third of participants are attackers.**

***Index Terms*—federated learning; distributed poisoning attack; defense; attack success rate; label-flipping**

I. Introduction

Comparing with centralized learning, federated learning [1] has advantages of distributed storing, computing, and private data preserving. In federated learning, participants (training terminals) report their local trained models rather than private training data, which protects the privacy of participants but makes the procedure of global model aggregating and updating more vulnerable to poisoning attacks. Poisoning attacks have been well explored in centralized and stand-alone settings, and the most commonly used poisoning strategies is *label-flipping* [2] in which an attacker modifies the label of training samples to another category, however the features of the data keep unchanged, which results in misclassification of classifiers attacked.

Several researchers have demonstrated the efficiency of single-attacker poisoning attacks in distributed learning scenarios. Bhagoji *et al.* [3] explore the threat of model poisoning attacks on federated learning launched by a single, non-colluding malicious participant. Hayes *et al.* [4] evaluate contamination attacks in multi-party machine learning. When the training set contains 5% contaminated data, the contamination attack success rate reaches 38%. Fung *et al.* consider the vulnerabilities of federated learning to sybil-based poisoning attacks [5]. However, few works investigate distributed poisoning attacks to federated learning in which multiple malicious participants inject poisoned training samples to the training procedures, and have the same attack goal. It is unclear whether distributed poisoning with multiple attackers is more effective than traditional poisoning with a single attacker or not (under the same number of poisoned training samples in total), and how the number of attackers and poisoned samples effect the attack success rate?

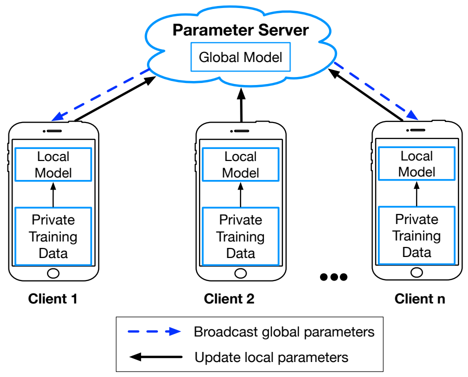


Fig. 1: The federated learning system architecture. The clients train models locally and share local model parameters to a server. The server aggregates model parameters of all clients, and then broadcasts the new global model to all clients.

In this paper, we build up a federated learning system, and implement the FederatedAveraging algorithm [1]. We launch distributed label-flipping poisoning attacks with different numbers of attackers and poisoned training samples, and examine the corresponding attack success rates. According to the experimental results, we obtain three observations about the relations between the number of attackers, poisoned samples, and attack success rate. First, the attack success rate increases linearly with the number of poisoned samples; second, increasing the number of attackers can improve the attack success rate without changing the total number of poisoned samples; third, the attack success rate increases faster with the number of poisoned samples when more attackers are involved. We measure the distance between models, and observe that in order to make the global model updated towards a wrong direction, attackers have to report poisoned local models which have large distance to honest ones. Thus, we propose a scheme,

Shan Chang is the corresponding author.

named Sniper, for the server to identify a set of honest local models, such that a clean global model can be aggregated by using it. In Sniper, the server constructs a graph, in which vertices refer to local models collected from participants during an updating, and if two local models are close enough (i.e., with relatively small Euclidean distance), then there exists an edge between them. Then, the sever identifies honest local models by solving a maximum clique problem in the graph. The global model is obtained by only aggregating those local models contained by the resulted clique. Experimental results show that the attack success rate drops to 2% even a third of participants are attackers under the protection of Sniper.

II. Federated Learning and Distributed Poisoning

A. Federated Learning

We consider a federated learning setup that consists of N participants O_k ($k \leq N$), each with access to a local private training dataset D_k, and a central parameter server S (illustrated in Fig. 1). At each epoch t, each participant O_k trains a new local model with local parameter vector $M_k^{(t)} \in \mathbb{R}^n$, where n is the dimensionality of the parameter space, and sends the model (i.e. the corresponding parameter vector) to S. When S receives $M_k^{(t)}$ ($k = 1, ..., N$) from all participants, it executes the FederatedAveraging algorithm [1] to obtain the new global model with global parameter vector $M_G^{(t)} \in \mathbb{R}^n$.

$$M_G^{(t)} = \frac{1}{N}\sum_{k=1}^{N} M_k^{(t)} \tag{1}$$

After receiving $M_G^{(t)}$ from S, each participant performs the following Stochastic Gradient Descent algorithm to generate a new local model.

$$M_k^{(t+1)} = M_G^{(t)} - \eta \cdot \bigtriangledown L(M_k^t, D_k) \tag{2}$$

where η, L, and $\bigtriangledown L$ represent learning rate, loss function and gradient, respectively.

B. Poisoning Attack

Deep learning needs a large amount of data which implies personal privacy information, for examples, home address, work place, religion and state of health. Not only releasing the data is harmful for the user, but also deliberately tainting data can cause serious accidents. For instance, the training dataset of a traffic signal sign recognize system is intentionally modified, and the autonomous vehicles use the system to discriminate the traffic sign. Although distort is minor, such as changing the label of the samples, it can cause shocking traffic accidents. Benjamin et al. [6] define this kind of manipulation as poisoning attack. They apply poisoning attack in anomaly detection, and the goal of attackers is to lead to a Denial of Service. Afterwards, poisoning attack is applied in SVM [2] and deep learning [7].

The poisoning attack scenarios can be classified into two categories:

Algorithm 1 Distributed Poisoning Attack in Federated Learning System

```
Require:
    $N$ participants indexed by $k$, training datasets $D_k$, local mini-batch size B, number of local epoch $E$, learning rate $\eta$, number of adversaries $P$, number of poisoned samples $S$.
Ensure:
    Parameter Server:
 1: initialize global model $M_G^0$
 2: for epoch $t$ $in$ $range$(0, E) do
 3:     for client $k$ $in$ $range$(N) do
 4:         $M_k^t \Leftarrow$ ClientShare($k$,$M_G^t$)
 5:     end for
        $M_G^{t+1} \Leftarrow \frac{1}{n}\sum_{k=1}^{n} M_k^t$
 6: end for
    ClientShare(k, $M_G$):
 7: //run on client k
 8: if $client\ k\ is\ adversries$ then
 9:     $D_a^k \Leftarrow$ poisoning $D_k$ with $\frac{S}{P}$ contaminated data //over-flipping
10: else
11:     $D_h^k \Leftarrow D_k$
12: end if
13: B $\Leftarrow$ randomly reorganize $D_h^k$ or $D_a^k$ into batches of size B
14: for batch $b \in$ [B] do
15:     $M_k \Leftarrow M_G - \eta \bigtriangledown L(M_k, b)$
16: end for
17: return $M_k$
```

Model failure poisoning attack. The goal of adversary is to make the model unavailable. In other words, the adversary's target is arbitrary, and their unanimous goal is to make the classifier give the wrong predictions.

Target error poisoning attack. This kind of poisoning attack is also to make the model's prediction wrong, but there is a difference that target error poisoning attack forces the model to misclassify a specific class (source label) to another target class (target label).

There are two main methods to generate poisoned samples (also called adversarial samples): label-flipping [2] and backdoor [9] [10] [11]. Label-flipping requires adversaries to change the labels of training data and keep the features of data unchanging. In backdoor attacks, adversaries modify a small region of the original training data to a secret pattern (white square or watermark) firstly and then relabeled it as target label. Actually, the pattern is a trigger making samples with it classified to target category.

There are some other method to contaminate data, such as adding noise to a panda image making the model predict it as gibbon, but they act on model testing phrase which is different from our attack goal [12] [13]. Further, label-flipping attack does not need any technology knowledge and pre-training. It is easier for an attacker to launch.

C. Distributed Poisoning

Distributed poisoning attack in federated learning system, as shown in Algorithm 1, refers to more than one attacker trains its own local models using the same loss function and hyper-parameters as honest participants as in Eq. 4 and Eq.

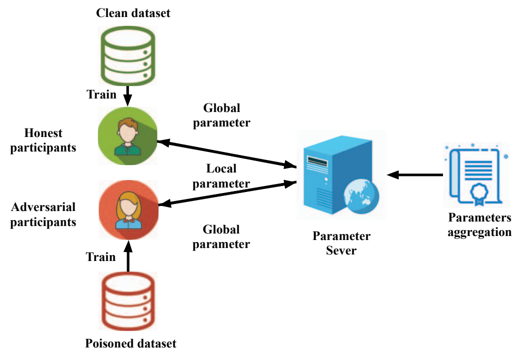


Fig. 2: Overview of the distributed poisoning federated learning system. Attackers pretend to be honest participants, and share poisoned parameters to the server. The server aggregates all parameters indiscriminately, and obtains a contaminated global model.

5, but poisoned training dataset (seen in Fig. 2). In this work, following assumptions are adopted:

- All attackers collude and have the same attack target. Particularly, they launch *Target error poisoning attack*.
- Attackers cannot observe the training data of other honest participants. This means that attackers cannot inference the parameters of any individual honest participant.
- The number of attackers should not exceed $N/3$.

Under an attack of distributed poisoning where $P(P \leq N/3)$ attackers hide among N participants, the global model aggregation algorithm on S can be re-written as Eq. 3

$$M_G^{(t)} = \frac{1}{N}(\sum_{h=1}^{N-P} M_h^{(t)} + \sum_{a=1}^{P} M_a^{(t)}) \tag{3}$$

where $M_h^{(t)}$ and $M_a^{(t)}$ indicate benign and poisoned local models, respectively, i.e.,

$$M_h^{(t)} = M_G^{(t)} - \eta \cdot \bigtriangledown L(M_h^{(t-1)}, D_h) \tag{4}$$

$$M_a^{(t)} = M_G^{(t)} - \eta \cdot \bigtriangledown L(M_a^{(t-1)}, D_a) \tag{5}$$

We define the success rate of attack as follows:

Definition 1. (**Attack Success Rate**) The attack succeeds if the poisoned model outputs the desired target label T for a source label I, otherwise the attack fails. The success rate SR of the attack for a model M is given as:

$$SR_M = \frac{n_T^{(I)}}{n^{(I)}} \times 100$$

where, $n^{(I)}$ indicates the number of testing samples with the source label I, and $n_T^{(I)}$ indicates the number of testing samples mislabeled as T.

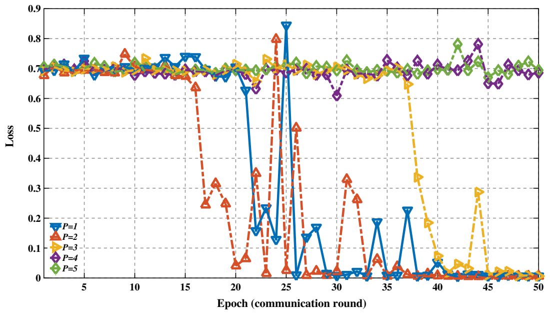


Fig. 3: Loss of the averaging algorithm. When the number of attackers is larger than 3, the loss does not decrease and the algorithm does not coverage.

TABLE I: Model architecture

Layer Type	# of Channels	Filter Size	Activation
Conv	32	5*5	ReLU
Conv	64	5*5	ReLU
Conv	128	5*5	ReLU
Conv	256	5*5	ReLU
FC	2	-	Softmax

III. Strategies of Distributed Poisoning

A. Experimental Setup

We evaluate the efficiency of distributed poisoning attacks on MNIST [15], a handwritten digit images dataset. Its training dataset contains 50,000 samples labeled as 0-9, and its testing dataset contains 10,000 samples. The size of the images in it is 28×28. According to the study of S. Shen *et al.*, launching targeted poisoning attacks on a classifier which distinguishes handwritten digits *0* to *9*, the easiest and hardest source and target label pairs are $(6, 2)$ and $(8, 4)$, respectively [16]. Thus, we set the following two attack goals in our experiments:

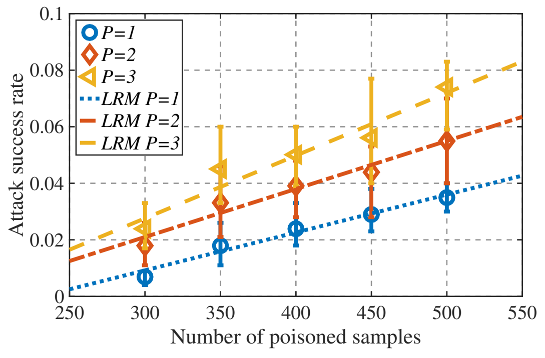
- Goal 1: given a digit image of *6*, the model outputs *2*.
- Goal 2: given a digit image of *8*, the model outputs *4*.

To eliminate the impact of other labels, we only use samples with the target and source labels to train a binary classifier (around 10,000 samples). We utilize *label-flipping* to generate poisoned samples, i.e., changing the labels of a training sample and keeping the feature of the sample unchanged, since it does not need any knowledge about the classifier and pre-training.

For the above datasets, we implement a federated learning system with *ten* participants. Training data are distributed to each participant evenly. Participants agree on the same model architecture and hyper-parameters. We choose a Convolutional Neural Network (CNN) (see the architecture in Table I), which has *four* convolution layers and *one* full connection layer, and uses *Relu* as the activate function for the first four layers and

TABLE II: Slope and intercept of the linear regression models

k, c / P	Goal 1		Goal 2	
	slope k	intercept c	slope k	intercept c
1	0.000134	-0.031	0.000073	-0.01065
2	0.00017	-0.03	0.000158	-0.02495
3	0.000222	-0.039	0.000208	-0.0141


(a) Goal 1. Source label: 6, target label: 2

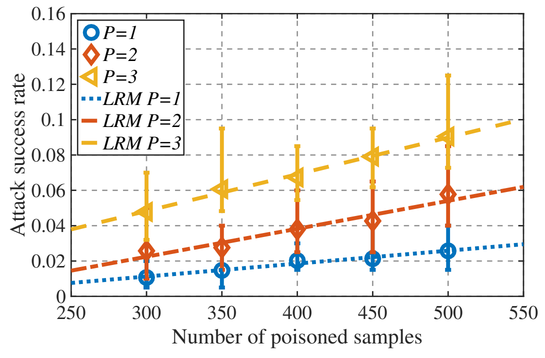
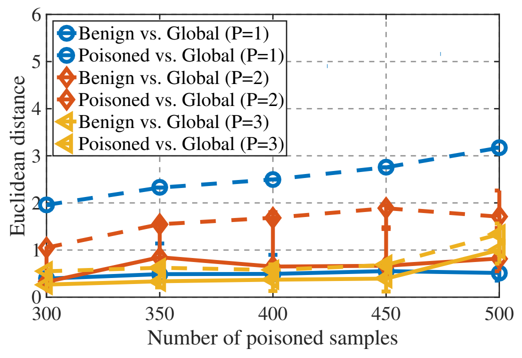

(b) Goal 2. Source label: 8, target label: 4

Fig. 4: Attack success rate vs. the number of poisoned samples


(a) Goal 1. Source label: 6, target label: 2

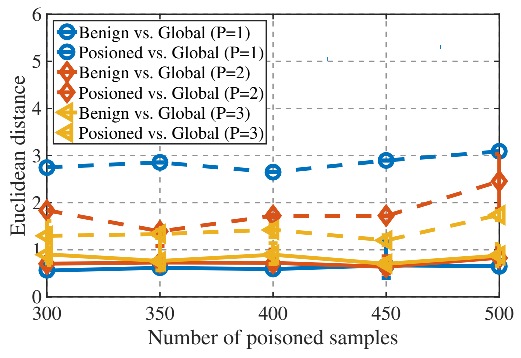

(b) Goal 2. Source label: 8, target label: 4

Fig. 5: Euclidean distances between local and global models vs. the number of poisoned samples

Softmax as the activate function for the output layer. We set the batch size B as *ten*. Learning rate η decays from 0.08 at an exponential rate of 0.098. Furthermore, we add a regularization term (λ=0.01) to prevent model overfitting. The weight of each neuron in the model is initialized to a random value sampled from a normal distribution $N(0, 0.02^2)$, and the bias is initialized to 0.

B. Baseline Federated Learning System

In the baseline system, all participants are honest, using only clean training samples to train a global model together. In an experiment, each participant randomly chooses 250 training samples with a target and a source label, respectively. We randomly choose 500 testing samples with the source label from the testing dataset. To evaluate the performance of baseline global model, we run the experiment for 20 times, and get the average. According to our experimental results, the accuracy of the global model is almost 100%. However, if a participant trains a local model without sharing parameters with others, the accuracy of the local model is around 90%. It indicates that the aggregation of local models does lead to a better global model.

C. Influence Factors on Distributed Poisoning Attacks

We consider the following two key factors effect the attack success rate of distributed poisoning attacks, i.e., the *number of poisoned samples* (over all attackers), and the *number of attackers*. Hence, we change the number of attackers P from 1 to 3, and under a certain number of attackers, we increase the total number of poisoned samples from 300 to 500 at an interval of 50, and distribute poisoned samples to all attackers evenly. We emphasize that the maximum number of attackers should be less than $1/3$ of the total number of participants [17]. Otherwise, the Federated Averaging algorithm does not coverage. As shown in Fig. 3, when number of adversaries is larger than 3, the loss of averaging algorithm does not decrease anymore.

We examine the attack success rates of goal 1 and 2, respectively, under different number of poisoned samples and attackers, which are shown in Fig. 4. (a) and (b). Furthermore, we consider the number of poisoned samples and the average attack success rate as independent (x) and dependent (y) variables, respectively. Considering a specific number of attackers, we conduct linear regression estimation based on the resulted average attack success rates under different numbers of poisoned samples. The estimated linear regression models

(LRM) $y = kx + c$ are illustrated in Table II, where k and c denoted as the slope and intercept, respectively. According to the above experiments and analysis, we make the following propositions:

Proposition 1. The attack success rate increases linearly with the number of poisoned samples.

Proposition 2. Increasing the number of attackers can improve the attack success rate without changing the number of poisoned samples.

Proposition 3. The slope of the LRM (i.e., increasing speed of attack success rate) increases with the number of attackers.

IV. Analyzing Distributed Poisoning

Given two models M_1 and M_2, we utilize Euclidean distance, which is one of the most common ways to measure distance in high dimensional space, to measure the difference between them (eq. 6).

$$Dis(M_1, M_2) = \sqrt{\sum (M_1 - M_2)^2} \tag{6}$$

Then, for each attack goal, we calculate the *average*, *maximum* and *minimum* Euclidean distances between benign local models and M_G, and between poisoned local models and M_G, respectively, under different numbers of poisoned samples and attackers, where M_G is the global model described in subsection II-C. According to the experimental results shown in Fig. 5 (a) and (b), we make the following analysis:

1) Impact of the number of poisoned samples: when increasing the number of poisoned samples, the deviation of updating from the new aggregated global model $M_G^{(t+1)}$ to a poisoned local model $M_{attacker}^{(t+1)}$, i.e., α in Fig. 6, will be decreased in general. It can be seen in Fig.5, for both attack goals, increasing the numbers of poisoned samples, the model distances between M_G^t and poisoned local modes $M_{attacker}^{(t+1)}$ will be increased, i.e., α will be enlarged, while the distances between M_G and benign local models remain stable. Consider that the global model update vector agg (black dotted vector) is a weighted sum vector of $honest$ and adv, as in Eq. 7, so it deviates from the correct objective direction by θ, i.e.,

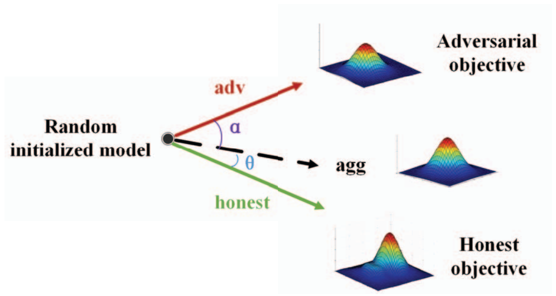


Fig. 6: Attackers have a different model updating direction (*adv*) from those honest persons, and the updating direction of global model (*agg*) is the aggregation of the two updating directions.

$$agg = \frac{N-P}{N} honst + \frac{P}{N} adv \tag{7}$$

More poisoned samples imply a large deviation of the poisoned global model from benign models. As a result, the aggregated vector agg derives away from the vector $honest$, which means a high attack success rate.

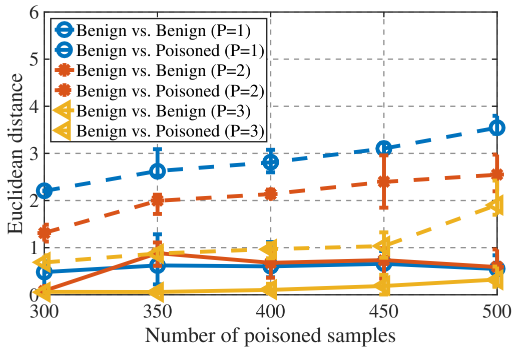
2) Impact of the number of attackers: for a certain number of poisoned samples, when the number of adversaries P increases, each attacker obtains fewer poisoned samples, thus the model distance from a poisoned model to M_G decreases. However, the efficiency of poisoned attacks improves (the attack success rate increases, see in Fig. 4), which is because more poisoned local models are involved in the global model aggregation. Furthermore, distributing poisoned samples to more attackers also means it is more difficult for the server to identify each poisoned local model.

Algorithm 2 Identify Honest Participansts

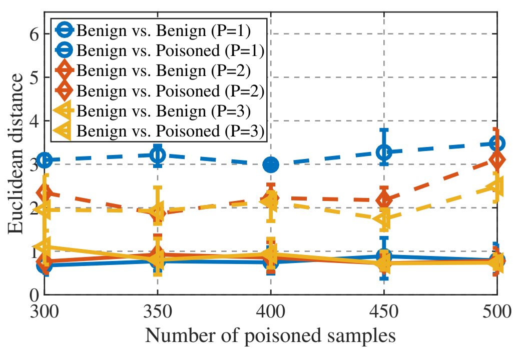
```
1:  $\gamma \Leftarrow 0.5$
2:  $\epsilon \Leftarrow 0.05$  //the length to decrease $\gamma$
3:  $Honest \Leftarrow \phi$  //$Honest$ is the set of honest models
4:  $V \Leftarrow \{M_1, M_2, ..., M_N\}$  //$V$ contains all the models
5:  while $Honest == \phi$ do
6:      //if distance between two models is less than $\gamma$, they are
        neighbors G
7:      for $i$ $in$ $range(N-1)$ do
8:          $G(M_i) \Leftarrow \phi$
9:          for $j$ $in$ $range(i+1, N)$ do
10:             if $Dis(M_i, M_j) < \gamma$ then
11:                 $G(M_i) \Leftarrow G(M_i) \cup M_j$
12:             end if
13:         end for
14:     end for
15:     $Cliques \Leftarrow BronKerbosch(V, G)$  //find all cliques
16:     //sort $Cliques$ by size and find the largest one
17:     $MaxClique \Leftarrow FindLargestClique(Cliques)$
18:     if $|MaxClique| > \frac{N}{2}$ then
19:         $Honest \Leftarrow MaxClique$
20:         return $Honest$
21:     end if
22:     $\gamma \Leftarrow \gamma + \epsilon$
23: end while
```

V. Defending Against Distributed Poisoning Attack

We notice that in order to force the aggregated global model deviating from the benign model, the poisoned local models have to be different from those benign local models trained by honest participants. As shown in Fig. 6, the green and red vectors, i.e., $honest$ and adv, present the benign and poisoned update direction of local models of honest participants and attackers. The angle between two vectors reflects the difference between two optimization objectives. Consider that attackers have completely different learning objectives from honest participants, we speculate that there exist big differences between benign and poisoned local models. Thus, we calculate the Euclidean distances between benign local models (from

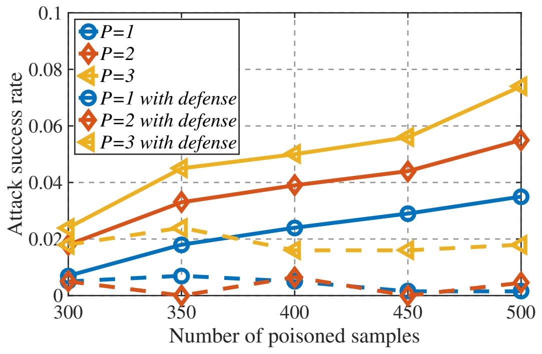


(a) Goal 1. Source label: 6, target label: 2

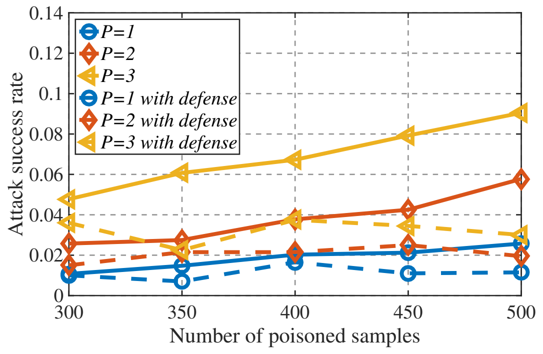


(b) Goal 2. Source label: 8, target label: 4

Fig. 7: Euclidean distances between local models vs. the number of poisoned samples



(a) Goal 1. Source label: 6, target label: 2



(b) Goal 2. Source label: 8, target label: 4

Fig. 8: Average attack success rate with and without defense vs. the number of poisoned samples

different honest participants), and between bengin and poisoned models, respectively. Fig. 7 demonstrates the *average*, *maximum* and *minimum* Euclidean distances under different numbers of attackers and poisoned samples. It can be seen that the Euclidean distances between benign and poisoned local models are significantly larger than that between benign local models.

Based on above analysis and observations, we propose a filtering mechanism, **Sniper**, which is conducted by the parameter server S, to remove attackers from the global model aggregation by examining Euclidean distances between local models. Now, we explain the mechanism in more detail:

- Firstly, after collecting local models M_i from N participants, S calculates Euclidean Distances between each pair of local models, i.e., $Dis(M_i, M_j)$.
- Second, S assigns an initial value to γ (we empirically set γ as *0.5* according to our experimental results shown in Fig. 7). Then, it constructs a graph such that each local model M_i refers to a vertex v_i in the graph, and if the Euclidean distance between M_i and M_j is smaller that the pre-determined threshold γ, there exists an edge $e_{i,j}$ between them.
- Third, S finds the maximum clique in the graph, and checks if the number of vertices in the clique is larger than $N/2$. If so, S aggregates vertices (local models) in the clique using Eq. 1, and gets the global model. Otherwise, increase γ with ϵ, and go to the second step.

The maximum clique problem is one of the classical NP-complete problems. We use BronKerbosch [18] to find all the cliques, and the largest one is the maximum clique(honest user candidate set), as shown in Algorithm 2.

S runs Sniper during every updating rather than only at the beginning of training. This can prevent attackers upload contaminated parameters at a few epochs or after a few benign sharing.

We evaluate the performance of Sniper in our federated learning system. We set ϵ as 0.05. The experimental results in Fig. 8 (a) and (b) show that the attack success rates drop significantly. In Goal 1, the attack success rate is close to 0. Even when three attackers are in the system in which Sniper filters more poisoned samples, the attack success rates is around 2% and 3% for attack goal 1 and 2, respectively. As a result, Sniper can distinguish honest users and attackers, and ignore models shared by attackers.

VI. Related Work

Distributed deep learning has been applied to actual scenarios. Liu et al. proposes a new method, called Federated-Autonomous Deep Learning (FADL) [19] which trains part of the model using all data sources in a distributed manner and other parts using data from specific data sources. This approach can be used to adapt the trained model to specific tasks as well as to partially protect data privacy.

Adversaries in collaborative learning and federated learning can be strong. They can inference the membership of other users' training set [20], but this attack limits the number of adversaries in the system is 1. Because the adversary only gets the global parameter, he cannot recognize the parameter of a specific user from the aggregated parameter. As a result, if there are multiply attackers, this attack cannot success. Bhagoji et al. [3] explore the threat of model poisoning attacks on federated learning initiated by a single, non-colluding malicious agent where the adversarial objective is to cause the model to misclassify a set of chosen inputs with high confidence.

Distributed learning is vulnerable. As shown in the experiment in this article, attackers just need to modify labels of their own local training dataset to degrade the global model performance. Therefore, a lot of defense methods for attacks in federated learning are proposed. Auror [16] detects malicious users and generates an accurate model based on the anomalous distribution of the masked features.

VII. Conclusion

In this paper we have analyzed how the number of poisoned samples and the number of attackers as variables affecting the performance of distributed poisoning attacks. The attack success rate increases linearly with the number of poisoned samples; The attack success rate increases with the number of attackers when the number of poisoned samples is unchanged and the increasing speed becomes faster when more attackers are involved.

Through observing the distance between models, we find models of honest users and attackers are in different cliques. Based on this we propose a filtering defense mechanism, Sniper. During every communication, parameter server runs it to filter parameters updated by attackers. As a result, Sniper can recognize honest users and drop attack success rate significantly even multiple attackers are in the federated learning system.

In the future, we will evaluate more aggregation algorithms in federated learning and analyze how multi-task poisoning attack attack global model. Further, we will expand the scale of federated learning by increasing the number of clients.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61972081, 61672151, 61772340, 61420106010), National Key Research and Development Project(Grant No. 2016QY12Z2103-2), Shanghai Rising-Star Program (Grant No.17QA1400100), National Key R&D Program of China (Grant No. 2018YFC1900700), Shanghai Municipal Natural Science Foundation (Grant No. 18ZR1401200), the Fundamental Research Funds for the Central Universities (Grant No. EG2018028), DHU Distinguished Young Professor Program and 2017 CCF-IFAA Research Fund.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[2] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*. Omnipress, 2012, pp. 1467–1474.

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

[4] J. Hayes and O. Ohrimenko, "Contamination attacks and mitigation in multi-party machine learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 6604–6615.

[5] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, 2018.

[6] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar, "Antidote: understanding and defending against poisoning of anomaly detectors," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*. ACM, 2009, pp. 1–14.

[7] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 27–38.

[8] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&amp;P)*. IEEE, 2016, pp. 372–387.

[9] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *arXiv preprint arXiv:1807.00459*, 2018.

[10] 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.

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

[12] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[13] P. Tabacof and E. Valle, "Exploring the space of adversarial images," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 426–433.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[15] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] S. Shen, S. Tople, and P. Saxena, "A uror: defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACM, 2016, pp. 508–519.

[17] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 119–129.

[18] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[19] D. Liu, T. Miller, R. Sayeed, and K. Mandl, "Fadl: Federated-autonomous deep learning for distributed electronic health record," *arXiv preprint arXiv:1811.11400*, 2018.

[20] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," *arXiv preprint arXiv:1805.04049*, 2018.