Poisoning Defense in Federated Learning: Attack Problem and Related Works

Qi Xiang Zhang
Institute of Information Management,
National Yang Ming Chiao Tung University, Taiwan

***Abstract*—This document describes the common poisoning attacks that appear in the learning process of federated learning. First, we will understand what federated learning is and its vulnerabilities. Second, we will focus on how attackers carry out poisoning attacks during the learning process of a global model. Finally, we will discuss four types of current defenses against poisoning attacks.**

***Index Terms*—Federated learning, cosine similarity, poisoning attack, blockchain, edge computing, security and protection, privacy.**

I. Introduction

Data silos refer to information that belongs exclusively to individual organizations, remains hidden from external access, and, as a result, lacks effective communication and collaboration between data silos, hindering comprehensive data analysis and utilization.

Traditional machine learning, which trains data sets centrally, encounters the problem of data silos. As problems become more complex, the sources of training data must become increasingly diverse. Therefore, cross-organizational data integration is necessary. To address the issue of data being inaccessible between different organizations, it is crucial to ensure data privacy when sharing information. This led to the emergence of a new machine learning training method known as federated learning.

In a typical federated learning framework, there is a centralized server and multiple clients. The learning process begins with the server providing a model to these clients, allowing them to train the model using their own local data. After training their respective models, these clients return their model parameters to the server. The server then updates the new model using predefined aggregation rules and provides the updated model to the clients for further training. This process continues until the model converges, marking the end of the training process. Throughout the entire training process, the local training data owned by the clients is not transmitted to the server, ensuring the privacy of the client-owned data.

A. Types of Federated Learning

The classification of federated learning into horizontally federated learning (HFL), vertically federated learning (VFL), and federated transfer learning (FTL) is based on [1], which categorizes federated learning according to the characteristics of the data owned by participants and the distribution of samples.

If the datasets of different participants have similar features, the federated learning falls into the category of HFL. HFL can further be divided into HFL to businesses (H2B) and HFL to consumers (H2C). H2B typically involves a smaller number of participants who often possess more computational power and a larger amount of data. On the other hand, H2C has a larger number of participants but lacks sufficient computational capacity and has less data.

If the datasets from the same participants appear repeatedly and have different data features, the federated learning belongs to VFL. This type typically occurs when different organizations have the same data records but with different field attributes.

If the datasets of participants in federated learning rarely overlap and have minimal shared data features, then the federated learning falls into the category of FTL.

B. Vulnerabilities of Federated Learning

Here's a brief introduction to the vulnerabilities of federated learning: Attackers in federated learning can come from either internal or external sources. Internal attackers include clients participating in model training and the server responsible for aggregating model parameters. External attackers encompass any external users who may eavesdrop on the communication channel between clients and the server.

The types of attacks on federated learning can be categorized into poisoning attacks and inference attacks. Poisoning attacks can be further divided into data poisoning attacks and model poisoning attacks. I will provide a brief introduction to these attacks:

- Data Poisoning Attack: In a data poisoning attack, an adversary injects malicious data or modifies genuine data within a participant's local dataset. These poisoned data points can influence the training process and potentially compromise the global model's integrity.
- Model Poisoning Attack: In a model poisoning attack, the attacker manipulates a participant's local model before aggregating it with others to form the global model. This can lead to the global model being controlled or influenced by the attacker.
- Inference Attack: In an inference attack, an adversary attempts to gain insights into a model's behavior or

extract sensitive information by making inferences from the global model's outputs.

These attacks pose significant security and privacy challenges in federated learning, and defense mechanisms are necessary to mitigate their impact. In the second part, our primary focus is on the attack type known as the poisoning attack in the context of federated learning. The more complex inference attack is not within the scope of our discussion.

II. The Attack Problem

In this paragraph, we will know real-life examples of poisoning attacks and the different types of poisoning attacks initiated by attackers for various purposes. We will also explore the differences and attack complexities between untargeted attacks and targeted attacks.

Untargeted attacks launched by attackers aim to reduce the overall accuracy of the federated learning model without targeting any specific data labels. In contrast, targeted attacks are different from untargeted attacks, as attackers specifically target the labels of certain data, causing the federated learning model to incorrectly identify a data's label as the target label. An actual example of a targeted attack is the label-flipping attack. In terms of the difficulty of execution, targeted attacks are generally more challenging to execute than untargeted attacks because they involve the attacker's specific objectives.

In the previous paragraphs, we learned that poisoning attacks can be divided into data poisoning attacks and model poisoning attacks. However, in reality, the differences between these two types are not substantial. In both types, malicious participants upload incorrect model parameters to the server after completing local model training, resulting in a significant decrease in the overall performance of federated learning models. If we were to emphasize a difference between the two, it would be that data poisoning attacks involve contaminated training data, whereas in model poisoning attacks, the training data remains uncontaminated, but participants maliciously adjust the model parameters before uploading them to the server.

Finally, let's illustrate a simple poisoning attack scenario. Suppose we want to train a model to recognize hand-written digits 0 to 9 using the MNIST dataset through federated learning. In this setup, we have a server responsible for providing the initial model and many clients for local training. However, among these clients, some are malicious attackers who aim to make the trained model misclassify hand-written digit 1 as digit 7. During the training process, these malicious clients use incorrectly labeled data for training. Since the server cannot actively filter out malicious model parameters uploaded by clients, such attacks are often successful. This type of attack falls under the category of target attacks, specifically a label-flipping attack. In the next paragraph, we will explore some defense methods that may be effective in countering poisoning attacks.

III. Related Works

Here are some methods for defending against poisoning attacks in federated learning. In this paragraph, we will discuss their operational principles and potential issues one by one.

A. FoolsGold [3]

A Sybil Attack occurs when a system allows users to freely enter and exit. Attackers can create a large number of malicious users and introduce them into the system, attempting to disrupt the normal operation of the system. Broadly speaking, a Sybil Attack can also be considered a type of poisoning attack.

FoolsGold is a novel method for countering Sybil Attacks. The authors adjust the learning rate of the client's parameter updates based on the similarity between the model updates uploaded by local clients. Since attackers often share the same attack objectives, the model updates they upload tend to be more similar than expected. By calculating the similarity of parameter updates, it becomes possible to eliminate attackers by adjusting the learning rate, without making assumptions about the actual number of attackers.

Here are some assumptions made by the authors during the design of the FoolsGold algorithm:

1) The server-side cannot be inherently malicious.
2) There must be at least one honest local client that updates the model parameters honestly.
3) The locally uploaded model parameter updates are not obfuscated, allowing the server-side to calculate the similarity of parameter updates.
4) The attackers' goal is to misclassify a specific class as a target class without affecting the recognition of other classes.
5) The federated learning model uses Stochastic Gradient Descent (SGD) as the optimization algorithm for parameter updates.

FoolsGold distinguishes between malicious and honest local clients by examining the similarity in the direction of model parameter updates. As mentioned earlier, attackers typically share a common attack goal, resulting in their uploaded model parameter update directions being quite similar. After calculating the update similarity, FoolsGold adjusts the learning rates based on the magnitude of the similarity. Local clients with higher update similarity will have smaller learning rates, reducing their influence on the overall model. FoolsGold considers both the current update similarity and historical update similarity information compared to other clients when adjusting the learning rates.

FoolsGold uses cosine similarity for calculating model parameter update similarity. This choice is made to avoid being influenced by the vector size, and when calculating cosine similarity, it selectively picks important indicator parameters, obtainable through the weight of the preceding layer in the classification layer. The resulting cosine similarity values fall within the range of -1 to 1. However, this approach presents potential issues. It cannot guarantee that high cosine similarity necessarily indicates malicious local clients. There is a risk of inadvertently penalizing honest local clients by reducing their learning rates. Moreover, the distribution of cosine similarity values may be too dispersed, making it challenging to identify malicious local clients. Therefore, FoolsGold introduces the Pardoning method to prevent the unintentional penalization of honest local clients. It achieves this by readjusting cosine

similarity, ensuring that at least one local client can assist in updating model parameters. Additionally, FoolsGold applies the logit function to transform the calculated cosine similarity values. The transformation concentrates the value distribution at both extremes, making it easier for FoolsGold to distinguish malicious local clients.

Before evaluating the pros and cons of the FoolsGold algorithm, it's important to provide additional context on the rules followed by the authors during its design:

1) When the system is not under attack, FoolsGold should not affect the performance of federated learning.
2) FoolsGold should reduce the contributions (by lowering the learning rates) of clients with similar parameter update directions.
3) FoolsGold should be resilient against increasing poisoning attacks launched by attackers.
4) FoolsGold should be able to distinguish between parameter updates that appear malicious but are actually from honest clients. (FoolsGold introduces "pardoning" to address misidentification issues and readjust the cosine similarity for honest clients.)
5) FoolsGold should not rely on specific assumptions about the clients and attackers, such as predefining the actual number of attackers.

With these rules in mind, we can now assess the advantages and disadvantages of the FoolsGold algorithm.

From the evaluation of FoolsGold presented by the authors in the paper, it's evident that FoolsGold struggles to handle scenarios where there is only one attacker. This difficulty arises because FoolsGold calculates cosine similarity pairwise between clients. When there is only one attacker, it becomes challenging to determine the appropriate cosine similarity without a reference point.

Furthermore, [2] points out that cosine similarity-based poisoning defense is fragile because attackers can specifically target a few neurons in a layer of the model, effectively evading cosine similarity calculations. This type of attack is known as a layer replacement attack (LRA).

These observations highlight potential limitations and vulnerabilities of the FoolsGold algorithm when dealing with single attackers and layer replacement attacks.

B. LoMar [4]

Local Malicious Factor (LoMar) is a two-stage defense method against poisoning attacks. The authors argue that existing defense methods only treat malicious updates as global anomalies in the Federated Learning (FL) system, without analyzing the feature of malicious remote updates in local trained model parameters. Therefore, they propose the LoMar algorithm with the aim of detecting abnormal parameter updates in Federated Learning from a local perspective rather than from the traditional global perspective.

LoMar's defense consists of two stages. The first stage involves calculating a malicious client coefficient, and the second stage determines a threshold to specify the range within which the malicious client coefficient should fall to be considered malicious. The LoMar defense method ultimately outputs a binary factor (0 or 1) to filter out parameter updates from malicious clients. If a client is determined to be malicious, the binary factor is set to 0, rendering the parameter updates uploaded by that client during local training invalid.

LoMar uses Kernel Density Estimation (KDE) to estimate the malicious client coefficient. In the preprocessing stage, parameter updates from remote clients are divided based on the dimensions of parameter features. Subsequently, the updates are grouped based on the output labels, and Kernel Density Estimation is performed on each output label. The Kernel Density Estimation values for each label are multiplied together, generating a numerical output. This output is then compared with the results of Kernel Density Estimation for parameter updates in the vicinity to calculate the malicious client coefficient.

While the authors of LoMar demonstrated in their paper, using ROC curves, that LoMar has a strong capability to detect malicious parameter updates compared to other poisoning attack defense algorithms, the process of calculating the malicious client coefficient is inherently uncertain. There are challenges, such as the choice of kernel function for KDE estimation, that introduce uncertainties. Moreover, in the typical federated learning framework, clients often upload only the minimal necessary model parameter updates. This means that estimating the actual malicious client coefficient using KDE may not always yield accurate results.

C. FLCert [5]

FLCert is an embedded federated learning framework designed to resist parameter updates from a certain number of malicious clients. The authors of this paper argue that existing defense methods against poisoning attacks fall into the categories of Byzantine-robust or malicious client detection. To address this issue, they introduce a new defense method called FLCert. In FLCert, clients participating in federated learning are first divided into groups. Using existing federated learning methods, clients within the same group learn the same global model. The output labels for the global model learned by clients within a group are determined through a voting process among the clients in that group.

FLCert offers two different client grouping approaches:

- FLCert-P, which involves random sampling of clients for grouping, with the possibility of duplicate selections.
- FLCert-D, where clients are grouped based on specific rules, and duplicate selections are not allowed.

FLCert introduces this novel approach to enhance the security and robustness of federated learning in the presence of malicious clients.

FLCert's success in defending against poisoning attacks from malicious clients can be attributed to its client grouping strategy. By dividing all clients into groups, malicious clients are effectively distributed among different groups. This dispersion results in a significant reduction in the overall impact of malicious clients on the global model. Furthermore, FLCert employs an approach where the final model parameter updates to be uploaded are determined through group voting. This means that unless the majority of clients within a group are

malicious (comprising over half of the total clients in that group), the uploaded model parameter updates will not be malicious. This design effectively safeguards the federated learning process against poisoning attacks.

Despite the successful defense against poisoning attacks from malicious clients through FLCert's client grouping strategy, it still faces some significant challenges. One of the issues is when there are too many groups, which results in a scarcity of clients within each group. During the training of the global model within each group, the available training data becomes increasingly limited. This approach may expedite the global model's training, but the ultimate performance of the aggregated global model may not be optimal. Another challenge is estimating the maximum number of tolerated malicious clients within each group. This issue arises in FLCert-P, which involves the possibility of repeatedly sampling clients for grouping. While the authors have proposed a theoretical framework for estimation, there is still some uncertainty surrounding the accuracy of this estimation in practice.

D. FLChain [6]

Blockchain is a special type of database, also known as a decentralized digital ledger, maintained by numerous nodes distributed across the globe. Blockchain data is organized into blocks, arranged in chronological order, and secured through cryptography. While blockchain technology was initially used for recording cryptocurrency transactions, it is equally suitable for recording various other types of digital data. Each block in a blockchain contains the hash value of the previous block, meaning that to alter any single block, all subsequent blocks must be modified. This task is highly challenging from a technical perspective, ensuring the immutability of the blockchain.

The authors of FLChain recognized that blockchain possesses features such as decentralization, immutability, and traceability. Consequently, they integrated federated learning with blockchain technology to create a framework called FLChain. Their aim is to effectively address the issues encountered by federated learning in edge computing, as listed below:

1) Users must place complete trust in the server responsible for aggregating model parameters.
2) The communication process for transmitting model update parameters from clients to the server is vulnerable.
3) Federated learning relies on a single server, and if that server is attacked, the entire federated learning training process comes to a halt.
4) A single server may not be capable of handling local data from millions of client devices.

However, while FLChain addresses some of the issues in federated learning, it also faces both internal and external threats. Internal threats include the possibility that the server responsible for aggregating gradient updates can reconstruct the original data based on these updates' gradient. Additionally, malicious clients may learn the data's structure from global model updates without the knowledge of other clients and the server. External threats involve attackers attempting to influence the training objectives of federated learning by modifying the features of training data or using contaminated training data. During the training process, attackers can exploit communication channels between the server and clients to attempt to obtain clients' personal information. Furthermore, external eavesdroppers may potentially control the aggregation process of model updates without the server's authorization.

Regarding the mentioned issues, FLChain's authors propose several potential solutions:

1) Data perturbation techniques can be employed to ensure the safety of data information, thereby encouraging users to participate in training. This is somewhat similar to data obfuscation, making it more challenging to reverse the original data.
2) Attack detection mechanisms can be integrated into the server responsible for aggregating model parameters. These mechanisms can assess the contribution weights of each client to filter out malicious clients.
3) Ensuring the security and privacy of wireless communication channels can be achieved through data encryption, communication authorization, and smart contracts on the blockchain.

These solutions aim to address the identified threats and enhance the overall security and privacy of FLChain.

While FLChain leverages many advantages of blockchain technology, it also introduces some of the existing drawbacks of blockchain. These include the high energy and time consumption associated with blockchain operations and the presence of various types of attacks against blockchain. Additionally, FLChain participants are required to download all historical model learning records, which poses a challenge in terms of storage resources. To attract more participants, a well-designed participation incentive mechanism is needed to harness the full potential of federated learning.

One limitation of this paper is that it presents the conceptual framework of FLChain but lacks real-world implementation with actual data. As a result, it's challenging to guarantee the effective operation of FLChain without empirical evidence.

In practical applications, it's essential to explore how FLChain performs and addresses its inherent challenges while considering the trade-offs between the benefits and limitations of blockchain technology in federated learning.

ACKNOWLEDGMENTS

This document was originally written in Chinese and translated into English using ChatGPT and Grammarly. The content is based on information from six papers related to federated learning, some of which have been cited hundreds of times. I appreciate the authors of these papers for providing me with a deeper understanding of the issues related to federated learning.

REFERENCES

[1] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020.

[2] H. Kasyap and S. Tripathy, "Hidden vulnerabilities in cosine similarity based poisoning defense," in *2022 56th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2022, pp. 263–268.
[3] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, 2018.
[4] X. Li, Z. Qu, S. Zhao, B. Tang, Z. Lu, and Y. Liu, "Lomar: A local defense against poisoning attack on federated learning," *IEEE Transactions on Dependable and Secure Computing*, 2021.
[5] X. Cao, Z. Zhang, J. Jia, and N. Z. Gong, "Flcert: Provably secure federated learning against poisoning attacks," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3691–3705, 2022.
[6] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.
[7] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1639–1654, 2022.

[1] [2] [3] [4] [5] [6] [7]

Qi Xiang Zhang received the BBA degree from the Department of Information Management, National Central University of Taiwan (NCU), in 2023. He is currently pursuing the MS degree with the Institute of Information Management, National Yang Ming Chiao Tung University of Taiwan, NYCU. His main research interests include data mining, machine learning security and privacy, and network security.