Threats to Federated Learning

Lingjuan Lyu[1(✉)], Han Yu[2], Jun Zhao[2], and Qiang Yang[3,4]

[1] National University of Singapore, Singapore, Singapore
lyulj@comp.nus.edu.sg
[2] School of Computer Science and Engineering,
Nanyang Technological University, Singapore, Singapore
{han.yu,junzhao}@ntu.edu.sg
[3] Department of AI, WeBank, Shenzhen, China
[4] Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Kowloon, Hong Kong
qyang@cse.ust.hk

Abstract. As data are increasingly being stored in different silos and societies becoming more aware of data privacy issues, the traditional centralized approach of training artificial intelligence (AI) models is facing strong challenges. Federated learning (FL) has recently emerged as a promising solution under this new reality. Existing FL protocol design has been shown to exhibit vulnerabilities which can be exploited by adversaries both within and outside of the system to compromise data privacy. It is thus of paramount importance to make FL system designers aware of the implications of future FL algorithm design on privacy-preservation. Currently, there is no survey on this topic. In this chapter, we bridge this important gap in FL literature. By providing a concise introduction to the concept of FL, and a unique taxonomy covering threat models and two major attacks on FL: 1) poisoning attacks and 2) inference attacks, we provide an accessible review of this important topic. We highlight the intuitions, key techniques as well as fundamental assumptions adopted by various attacks, and discuss promising future research directions towards more robust privacy preservation in FL.

Keywords: Federated learning · Attacks · Privacy · Robustness

1 Introduction

As computing devices become increasingly ubiquitous, people generate huge amounts of data through their day to day usage. Collecting such data into centralized storage facilities is costly and time consuming. Another important concern is data privacy and user confidentiality as the usage data usually contain sensitive information [1]. Sensitive data such as facial images, location-based services, or health information can be used for targeted social advertising and recommendation, posing the immediate or potential privacy risks. Hence, private data should not be directly shared without any privacy consideration. As societies become increasingly aware of privacy preservation, legal restrictions such

Q. Yang et al. (Eds.): Federated Learning, LNAI 12500, pp. 3–16, 2020.
https://doi.org/10.1007/978-3-030-63076-8_1

Table 1. Taxonomy for horizontal federated learning (HFL).

HFL	Number of participants	FL training participation	Technical capability
H2B	Small	Frequent	High
H2C	Large	Not frequent	Low

as the General Data Protection Regulation (GDPR) are emerging which makes data aggregation practices less feasible [48].

Traditional centralized machine learning (ML) cannot support such ubiquitous deployments and applications due to infrastructure shortcomings such as limited communication bandwidth, intermittent network connectivity, and strict delay constraints [26]. In this scenario, federated learning (FL) which pushes model training to the devices from which data originate emerged as a promising alternative ML paradigm [35]. FL enables a multitude of participants to construct a joint ML model without exposing their private training data [12,35]. It can handle unbalanced and non-independent and identically distributed (non-IID) data which naturally arise in the real world [34]. In recent years, FL has benefited a wide range of applications such as next word prediction [34,36], visual object detection for safety [29], etc.

1.1 Types of Federated Learning

Based on the distribution of data features and data samples among participants, federated learning can be generally classified as horizontally federated learning (HFL), vertically federated learning (VFL) and federated transfer learning (FTL) [47].

Under HFL, datasets owned by each participant share similar features but concern different users [24]. For example, several hospitals may each store similar types of data (e.g., demographic, clinical, and genomic) about different patients. If they decide to build a machine learning model together using FL, we refer to such a scenario as HFL. In this chapter, we further classify HFL into HFL to businesses (H2B), and HFL to consumers (H2C). A comparison between H2B and H2C is listed in Table 1. The main difference lies in the number of participants, FL training participation level, and technical capability, which can influence how adversaries attempt to compromise the FL system. Under H2B, there are typically a handful of participants. They can be frequently selected during FL training. The participants tend to possess significant computational power and sophisticated technical capabilities [48]. Under H2C, there can be thousands or even millions of potential participants. In each round of training, only a subset of them are selected. As their datasets tend to be small, the chance of a participant being selected repeatedly for FL training is low. They generally possess limited computational power and low technical capabilities. An example of H2C is Google's GBoard application [36].

VFL is applicable to the cases in which participants have large overlaps in the sample space but differ in the feature space, *i.e.*, different participants

hold different attributes of the same records [46]. VFL mainly targets business participants. Thus, the characteristics of VFL participants are similar to those of H2B participants.

FTL deals with scenarios in which FL participants have little overlap in both the sample space and the feature space [48]. Currently, there is no published research studying threats to FTL models.

1.2 Threats to FL

FL offers a privacy-aware paradigm of model training which does not require data sharing and allows participants to join and leave a federation freely. Nevertheless, recent works have demonstrated that FL may not always provide sufficient privacy guarantees, as communicating model updates throughout the training process can nonetheless reveal sensitive information [8,37] even incur deep leakage [52], either to a third-party, or to the central server [2,36]. For instance, as shown by [3], even a small portion of gradients may reveal information about local data. A more recent work showed that the malicious attacker can completely steal the training data from gradients in a few iterations [52].

FL protocol designs may contain vulnerabilities for both (1) the (potentially malicious) server, who can observe individual updates over time, tamper with the training process and control the view of the participants on the global parameters; and (2) any participant who can observe the global parameter, and control its parameter uploads. For example, malicious participants can deliberately alter their inputs or introduce stealthy backdoors into the global model. Such attacks pose significant threats to FL, as in centralized learning only the server can violate participants' privacy, but in FL, any participant may violate the

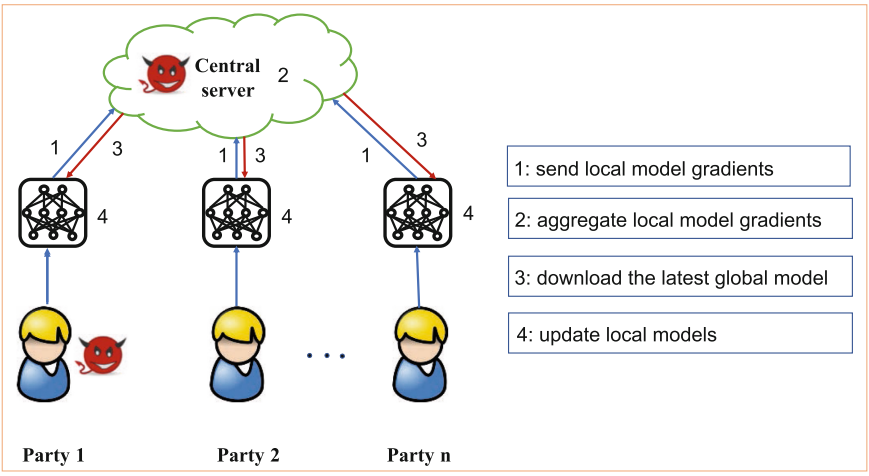

Fig. 1. A typical FL training process, in which both the (potentially malicious) FL server/aggregator and malicious participants may compromise the FL system.

privacy of other participants in the system, even without involving the server. Therefore, it is important to understand the principles behind these attacks. Existing survey papers on FL mostly focused on the broad aspect of how to make FL work [23,27,47]. In this chapter, we survey recent advances in threats to compromise FL to bridge this important gap in the artificial intelligence (AI) research community's understanding in this topic. In particular, we focus on two specific threats initiated by the insiders on FL systems: 1) poisoning attacks that attempt to prevent a model from being learned at all, or to bias the model to produce inferences that are preferable to the adversary; and 2) inference attacks that target participant privacy. The properties of these attacks are summarized in Table 2.

Table 2. A summary of attacks against server-based FL.

Attack type	Attack targets		Attacker role		FL scenario		Attack complexity		
	Model	Training data	Participant	Server	H2B	H2C	Attack iteration		Auxiliary knowledge required
							One round	Multiple rounds	
Data poisoning	YES	NO	YES	NO	YES	YES	YES	YES	YES
Model poisoning	YES	NO	YES	NO	YES	NO	YES	YES	YES
Infer class representatives	NO	YES	YES	YES	YES	NO	NO	YES	YES
Infer membership	NO	YES	YES	YES	YES	NO	NO	YES	YES
Infer properties	NO	YES	YES	YES	YES	NO	NO	YES	YES
Infer training inputs and labels	NO	YES	NO	YES	YES	NO	NO	YES	NO

2 Threat Models

Before reviewing attacks on FL, we first present a summary of the threat models.

2.1 Insider v.s. Outsider

Attacks can be carried out by insiders and outsiders. Insider attacks include those launched by the FL server and the participants in the FL system. Outsider attacks include those launched by the eavesdroppers on the communication channel between participants and the FL server, and by users of the final FL model when it is deployed as a service.

Insider attacks are generally stronger than the outsider attacks, as it strictly enhances the capability of the adversary. Due to this stronger behavior, our discussion of attacks against FL will focus primarily on the insider attacks, which can take one of the following three general forms:

1. Single attack: a single, non-colluding malicious participant aims to cause the model to miss-classify a set of chosen inputs with high confidence [4,7];
2. Byzantine attack: the byzantine malicious participants may behave completely arbitrarily and tailor their outputs to have similar distribution as the correct model updates, making them difficult to detect [11,14,15,40,49];
3. Sybil attack: the adversaries can simulate multiple dummy participant accounts or select previously compromised participants to mount more powerful attacks on FL [4,17].

2.2 Semi-honest v.s. Malicious

Under the semi-honest setting, adversaries are considered passive or honest-but-curious. They try to learn the private states of other participants without deviating from the FL protocol. The passive adversaries are assumed to only observe the aggregated or averaged gradient, but not the training data or gradient of other honest participants. Under the malicious setting, an active, or malicious adversary tries to learn the private states of honest participants, and deviates arbitrarily from the FL protocol by modifying, re-playing, or removing messages. This strong adversary model allows the adversary to conduct particularly devastating attacks.

2.3 Training Phase v.s. Inference Phase

Attacks at training phase attempt to learn, influence, or corrupt the FL model itself [9]. During training phase, the attacker can run data poisoning attacks to compromise the integrity of training dataset collection, or model poisoning attacks to compromise the integrity of the learning process. The attacker can also launch a range of inference attacks on an individual participant's update or on the aggregate of updates from all participants.

Attacks at inference phase are called evasion/exploratory attacks [5]. They generally do not tamper with the targeted model, but instead, either cause it to produce wrong outputs (targeted/untargeted) or collect evidence about the model characteristics. The effectiveness of such attacks is largely determined by the information that is available to the adversary about the model. Inference phase attacks can be classified into white-box attacks (i.e. with full access to the FL model) and black-box attacks (i.e. only able to query the FL model). In FL, the model maintained by the server not only suffers from the same evasion attacks as in the general ML setting when the target model is deployed as a service, the model broadcast step in FL renders the model accessible to any malicious client. Thus, FL requires extra efforts to defend against white-box evasion attacks. In this survey, we omit the discussion of inference phase attacks, and mainly focus on the training phase attacks.

3 Poisoning Attacks

Depending on the attacker's objective, poisoning attacks can be either a) random attacks and b) targeted attacks [22]. Random attacks aim to reduce the accuracy

of the FL model, whereas targeted attacks aim to induce the FL model to output the target label specified by the adversary. Generally, targeted attacks is more difficult than random attacks as the attacker has a specific goal to achieve. Poisoning attacks during the training phase can be performed on the data or on the model. Figure 2 shows that the poisoned updates can be sourced from two poisoning attacks: (1) data poisoning attack during local data collection; and (2) model poisoning attack during local model training process. At a high level, both poisoning attacks attempt to modify the behavior of the target model in some undesirable way. If adversaries can compromise the FL server, then they can easily perform both targeted and untargeted poisoning attacks on the trained model.

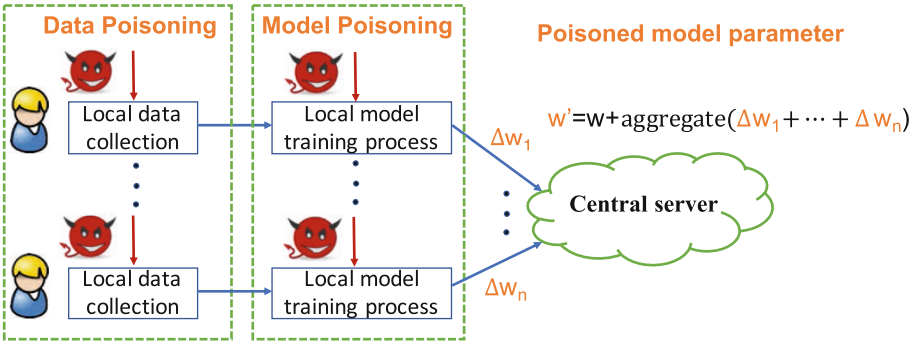

Fig. 2. Data v.s. model poisoning attacks in FL.

3.1 Data Poisoning

Data poisoning attacks largely fall in two categories: 1) clean-label [42] and 2) dirty-label [19]. Clean-label attacks assume that the adversary cannot change the label of any training data as there is a process by which data are certified as belonging to the correct class and the poisoning of data samples has to be imperceptible. In contrast, in dirty-label poisoning, the adversary can introduce a number of data sample it wishes to miss-classify with the desired target label into the training set.

One common example of dirty-label poisoning attack is the label-flipping attack [10,17]. The labels of honest training examples of one class are flipped to another class while the features of the data are kept unchanged. For example, the malicious participants in the system can poison their dataset by flipping all 1 s into 7 s. A successful attack produces a model that is unable to correctly classify 1 s and incorrectly predicts them to be 7 s. Another weak but realistic attack scenario is backdoor poisoning [19]. Here, an adversary can modify individual features or small regions of the original training dataset to embed backdoors into the model, so that the model behaves according to the adversary's objective if the input contains the backdoor features (*e.g.*, a stamp on an image).

However, the performance of the poisoned model on clean inputs is not affected. In this way, the attacks are harder to be detected.

Data poisoning attacks can be carried out by any FL participant. The impact on the FL model depends on the extent to which participants in the system engage in the attacks, and the amount of training data being poisoned. Data poisoning is less effective in settings with fewer participants like H2C.

3.2 Model Poisoning

Model poisoning attacks aim to poison local model updates before sending them to the server or insert hidden backdoors into the global model [4].

In targeted model poisoning, the adversary's objective is to cause the FL model to miss-classify a set of chosen inputs with high confidence. Note that these inputs are not modified to induce miss-classification at test time as under adversarial example attacks [45]. Rather, the miss-classification is a result of adversarial manipulations of the training process. Recent works have investigated poisoning attacks on model updates in which a subset of updates sent to the server at any given iteration are poisoned [7,11]. These poisoned updates can be generated by inserting hidden backdoors, and even a single-shot attack may be enough to introduce a backdoor into a model [4].

Bhagoji *et al.* [7] demonstrated that model poisoning attacks are much more effective than data poisoning in FL settings by analyzing a targeted model poisoning attack, where a single, non-colluding malicious participant aims to cause the model to miss-classify a set of chosen inputs with high confidence. To increase attack stealth and evade detection, they use the alternating minimization strategy to alternately optimize for the training loss and the adversarial objective, and use parameter estimation for the benign participants' updates. This adversarial model poisoning attack can cause targeted poisoning of the FL model undetected.

In fact, model poisoning subsumes data poisoning in FL settings, as data poisoning attacks eventually change a subset of updates sent to the model at any given iteration [17]. This is functionally identical to a centralized poisoning attack in which a subset of the whole training data is poisoned. Model poisoning attacks require sophisticated technical capabilities and high computational resources. Such attacks are generally less suitable for H2C scenarios, but more likely to happen in H2B scenarios.

4 Inference Attacks

Exchanging gradients in FL can result in serious privacy leakage [37,41,44,52]. As illustrated in Fig. 3, model updates can leak extra information about the unintended features about participants' training data to the adversarial participants, as deep learning models appear to internally recognize many features of the data that are not apparently related with the main tasks. The adversary can

also save the snapshot of the FL model parameters, and conduct property inference by exploiting the difference between the consecutive snapshots, which is equal to the aggregated updates from all participants less the adversary (Fig. 4).

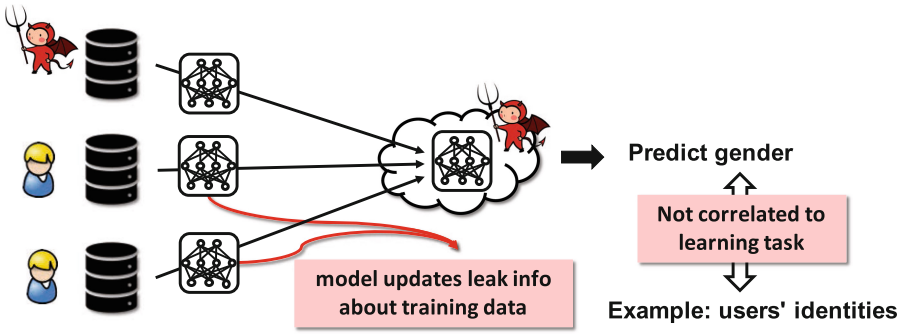

Fig. 3. Attacker infers information unrelated to the learning task.

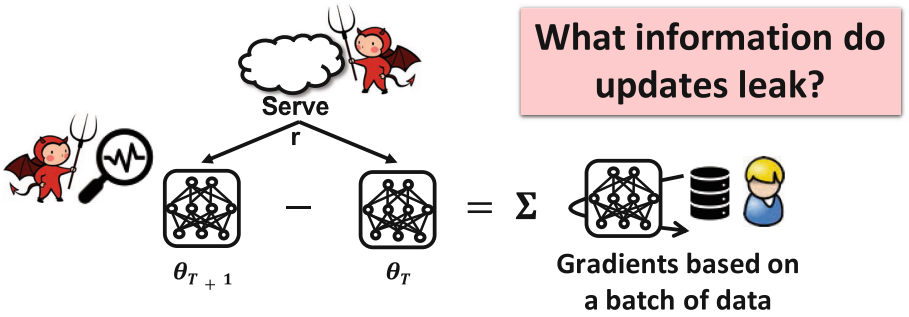

Fig. 4. Attacker infers gradients from a batch of training data.

The main reason is that the gradients are derived from the participants' private data. In deep learning models, gradients of a given layer are computed using this layer's features and the error from the layer above. In the case of sequential fully connected layers, the gradients of the weights are the inner products of the error from the layer above and the features. Similarly, for a convolutional layer, the gradients of the weights are convolutions of the error from the layer above and the features [37]. Consequently, observations of model updates can be used to infer a significant amount of private information, such as class representatives, membership as well as properties associated with a subset of the training data. Even worse, an attacker can infer labels from the shared gradients and recover the original training samples without requiring any prior knowledge about the training set [52].

4.1 Inferring Class Representatives

Hitaj *et al.* [21] devised an active inference attack called *Generative Adversarial Networks* (GAN) attack on deep FL models. Here, a malicious participant can intentionally compromise any other participant. The GAN attack exploits the real-time nature of the FL learning process that allows the adversarial participant to train a GAN that generates prototypical samples of the targeted training data which were meant to be private. The generated samples appear to come from the same distribution as the training data. Hence, GAN attack is not targeted at reconstructing actual training inputs, but only class representatives. It should be noted that GAN attack assumes that the entire training corpus for a given class comes from a single participant, and only in the special case where all class members are similar, GAN-constructed representatives are similar to the training data. This resembles model inversion attacks in the general ML settings [16]. However, these assumptions may be less practical in FL. Moreover, GAN attack is less suitable for H2C scenarios, as it requires large computation resources.

4.2 Inferring Membership

Given an exact data point, membership inference attacks aim to determine if it was used to train the model [43]. For example, an attacker can infer whether a specific patient profile was used to train a classifier associated with a disease. FL presents interesting new avenues for such attacks. In FL, the adversary's objective is to infer if a particular sample belongs to the private training data of a single participant (if target update is of a single participant) or of any participant (if target update is the aggregate). For example, the non-zero gradients of the embedding layer of a deep learning model trained on natural-language text reveal which words appear in the training batches used by the honest participants during FL model training. This enables an adversary to infer whether a given text appeared in the training dataset [37].

Attackers in an FL system can conduct both active and passive membership inference attacks [37,38]. In the passive case, the attacker simply observes the updated model parameters and performs inference without changing anything in the local or global collaborative training procedure. In the active case, however, the attacker can tamper with the FL model training protocol and perform a more powerful attack against other participants. Specifically, the attacker shares malicious updates and forces the FL model to share more information about the participants' local data the attacker is interested in. This attack, called gradient ascent attack [38], exploits the fact that SGD optimization updates model parameters in the opposite direction of the gradient of the loss.

4.3 Inferring Properties

An adversary can launch both passive and active property inference attacks to infer properties of other participants' training data that are independent of the features that characterize the classes of the FL model [37]. Property inference attacks assume that the adversary has auxiliary training data correctly

labelled with the property he wants to infer. An passive adversary can only observe/eavesdrop the updates and perform inference by training a binary property classifier. An active adversary can use multi-task learning to trick the FL model into learning a better separation for data with and without the property, and thus extract more information. An adversarial participant can even infer when a property appears and disappears in the data during training (e.g., identifying when a person first appears in the photos used to train a gender classifier). The assumption in property inference attacks may prevent its applicability in H2C.

4.4 Inferring Training Inputs and Labels

The most recent work called Deep Leakage from Gradient (DLG) proposed an optimization algorithm that can obtain both the training inputs and the labels in just a few iterations [52]. This attack is much stronger than previous approaches. It can recover pixel-wise accurate original images and token-wise matching original texts. [50] presented an analytical approach called Improved Deep Leakage from Gradient (iDLG), which can certainly extract labels from the shared gradients by exploiting the relationship between the labels and the signs of corresponding gradients. iDLG is valid for any differentiable model trained with cross-entropy loss over one-hot labels, which is the general case for classification.

Inference attacks generally assume that the adversaries possess sophisticated technical capabilities and large computational resources. In addition, adversaries must be selected for many rounds of FL training. Thus, it is not suitable for H2C scenarios, but more likely under H2B scenarios. Such attacks also highlight the need for protecting the gradients being shared during FL training, possibly through mechanisms such as homomorphic encryption [48].

5 Discussions and Promising Directions

There are still potential vulnerabilities which need to be addressed in order to improve the robustness of FL systems. In this section, we outline research directions which we believe are promising.

Curse of Dimensionality: Large models, with high dimensional parameter vectors, are particularly susceptible to privacy and security attacks [13]. Most FL algorithms require overwriting the local model parameters with the global model. This makes them susceptible to poisoning and backdoor attacks, as the adversary can make small but damaging changes in the high-dimensional models without being detected. Thus, sharing model parameters may not be a strong design choice in FL, it opens all the internal state of the model to inference attacks, and maximizes the model's malleability by poisoning attacks. To address these fundamental shortcomings of FL, it is worthwhile to explore whether sharing model updates is essential. Instead, sharing less sensitive information (e.g., SIGNSGD [6]) or only sharing model predictions [13] in a black-box manner may result in more robust privacy protection in FL.

Vulnerabilities to Free-Riding Participants: In FL system, there may exist free-riders in the collaborative learning system, who aim to benefit from the global model, but do not want to contribute any real information. The main incentives for free-rider to submit fake information may include: (1) one participant may not have any data to train a local model; (2) one participant is too concerned about its privacy to release any information that may compromise privacy; (3) one participant may not want to consume any local computation power to train any model [32,33]. In the current FL paradigm [34], all participants receive the same federated model at the end of collaborative model training regardless of their contributions. This makes the paradigm vulnerable to free-riding participants [28,32,33].

Threats to VFL: In VFL [20], there may only be one participant who owns labels for the given learning task. It is unclear if all the participants have equal capability of attacking the FL model, and if threats to HFL can work on VFL. Most of the current threats still focus on HFL. Thus, threats on VFL, which is important to businesses, are worth exploring.

FL with Heterogeneous Architectures: Sharing model updates is typically limited only to homogeneous FL architectures, *i.e.*, the same model is shared with all participants. It would be interesting to study how to extend FL to collaboratively train models with heterogeneous architectures [13,18,25], and whether existing attacks and privacy techniques can be adapted to this paradigm.

Decentralized Federated Learning: Decentralized FL where no single server is required in the system is currently being studied [32,33,36,48]. This is a potential learning framework for collaboration among businesses which do not trust any third party. In this paradigm, each participant could be elected as a server in a round robin manner. It would be interesting to investigate if existing threats on server-based FL still apply in this scenario. Moreover, it may open new attack surfaces. One possible example is that the last participant who was elected as the server is more likely to effectively contaminate the whole model if it chooses to insert backdoors. This resembles the fact in server-based FL models which are more vulnerable to backdoors in later rounds of training nearing convergence. Similarly, if decentralized training is conducted in a "ring all reduce" manner, then any malicious participant can steal the training data from its neighbors.

Weakness of Current Defense: FL with secure aggregation are especially susceptible to poisoning attacks as the individual updates cannot be inspected. It is still unclear if adversarial training can be adapted to FL, as adversarial training was developed primarily for IID data, and it is still a challenging problem how it performs in non-IID settings. Moreover, adversarial training typically requires many epochs, which may be impractical in H2C. Another possible defense is based on differential privacy (DP) [30–33,36,51]. Record-level DP bounds the success of membership inference, but does not prevent property inference applied to a group of training records [37]. Participant-level DP, on the other hand, is geared to work with thousands of users for training to converge and achieving an acceptable trade-off between privacy and accuracy [36]. The FL model fails

to converge with a small number of participants, making it unsuitable for H2B scenarios. Furthermore, DP may hurt the accuracy of the learned model [39], which is not appealing to the industry. Further work is needed to investigate if participant-level DP can protect FL systems with few participants.

Optimizing Defense Mechanism Deployment: When deploying defense mechanisms to check if any adversary is attacking the FL system, the FL server will need to incur extra computational cost. In addition, different defense mechanisms may have different effectiveness against various attacks, and incur different cost. It is important to study how to optimize the timing of deploying defense mechanisms or the announcement of deterrence measures. Game theoretic research holds promise in addressing this challenge.

Federated learning is still in its infancy and will continue to be an active and important research area for the foreseeable future. As FL evolves, so will the attack mechanisms. It is of vital importance to provide a broad overview of current attacks on FL so that future FL system designers are aware of the potential vulnerabilities in their designs. This survey serves as a concise and accessible overview of this topic, and it would greatly help our understanding of the threat landscape in FL. Global collaboration on FL is emerging through a number of workshops in leading AI conferences[1]. The ultimate goal of developing a general purpose defense mechanism robust against various attacks without degrading model performance will require interdisciplinary effort from the wider research community.

References

1. Abadi, M., et al.: Deep learning with differential privacy. In: CCS, pp. 308–318 (2016)
2. Agarwal, N., Suresh, A.T., Yu, F.X.X., Kumar, S., McMahan, B.: cpSGD: communication-efficient and differentially-private distributed SGD. In: NeurIPS, pp. 7564–7575 (2018)
3. Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. **13**(5), 1333–1345 (2018)
4. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. CoRR, arXiv:1807.00459 (2018)
5. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: ICCS, pp. 16–25 (2006)
6. Bernstein, J., Zhao, J., Azizzadenesheli, K., Anandkumar, A.: signSGD with majority vote is communication efficient and fault tolerant. CoRR, arXiv:1810.05291 (2018)
7. Bhagoji, A.N., Chakraborty, S., Mittal, P., Calo, S.: Analyzing federated learning through an adversarial lens. CoRR, arXiv:1811.12470 (2018)
8. Bhowmick, A., Duchi, J., Freudiger, J., Kapoor, G., Rogers, R.: Protection against reconstruction and its applications in private federated learning. CoRR, arXiv:1812.00984 (2018)

[1] https://www.ntu.edu.sg/home/han.yu/FL.html.

9. Biggio, B., Nelson, B., Laskov, P.: Support vector machines under adversarial label noise. In: ACML, pp. 97–112 (2011)
10. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. CoRR, arXiv:1206.6389 (2012)
11. Blanchard, P., Guerraoui, R., Stainer, J., et al.: Machine learning with adversaries: Byzantine tolerant gradient descent. In: NeurIPS, pp. 119–129 (2017)
12. Bonawitz, K., et al.: Practical secure aggregation for privacy-preserving machine learning. In: CCS, pp. 1175–1191 (2017)
13. Chang, H., Shejwalkar, V., Shokri, R., Houmansadr, A.: Cronus: robust and heterogeneous collaborative learning with black-box knowledge transfer. CoRR, arXiv:1912.11279 (2019)
14. Chen, L., Wang, H., Charles, Z., Papailiopoulos, D.: Draco: Byzantine-resilient distributed training via redundant gradients. CoRR, arXiv:1803.09877 (2018)
15. Chen, Y., Su, L., Xu, J.: Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. Proc. ACM Meas. Anal. Comput. Syst. **1**(2), 44 (2017)
16. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: CCS, pp. 1322–1333 (2015)
17. Fung, C., Yoon, C.J., Beschastnikh, I.: Mitigating sybils in federated learning poisoning. CoRR, arXiv:1808.04866 (2018)
18. Gao, D., Liu, Y., Huang, A., Ju, C., Yu, H., Yang, Q.: Privacy-preserving heterogeneous federated transfer learning. In: IEEE BigData (2019)
19. Gu, T., Dolan-Gavitt, B., Garg, S.: BadNets: identifying vulnerabilities in the machine learning model supply chain. CoRR, arXiv:1708.06733 (2017)
20. Hardy, S., et al.: Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. CoRR, arXiv:1711.10677 (2017)
21. Hitaj, B., Ateniese, G., Pérez-Cruz, F.: Deep models under the GAN: information leakage from collaborative deep learning. In: CSS, pp. 603–618 (2017)
22. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.D.: Adversarial machine learning. In: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, pp. 43–58 (2011)
23. Kairouz, P., et al.: Advances and open problems in federated learning. CoRR, arXiv:1912.04977 (2019)
24. Kantarcioglu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Trans. Knowl. Data Eng. **16**(9), 1026–1037 (2004)
25. Li, D., Wang, J.: FedMD: heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581 (2019)
26. Li, H., Ota, K., Dong, M.: Learning IoT in edge: deep learning for the Internet of Things with edge computing. IEEE Netw. **32**(1), 96–101 (2018)
27. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: challenges, methods, and future directions. CoRR, arXiv:1908.07873 (2019)
28. Lingjuan Lyu, X.X., Wang, Q.: Collaborative fairness in federated learning. arxiv.org/abs/2008.12161v1 (2020)
29. Liu, Y., et al.: Fedvision: an online visual object detection platform powered by federated learning. In: IAAI (2020)
30. Lyu, L., Bezdek, J.C., He, X., Jin, J.: Fog-embedded deep learning for the Internet of Things. IEEE Trans. Ind. Inform. **15**(7), 4206–4215 (2019)

31. Lyu, L., Bezdek, J.C., Jin, J., Yang, Y.: FORESEEN: towards differentially private deep inference for intelligent Internet of Things. IEEE J. Sel. Areas Commun. **38**, 2418–2429 (2020)
32. Lyu, L., Li, Y., Nandakumar, K., Yu, J., Ma, X.: How to democratise and protect AI: fair and differentially private decentralised deep learning. IEEE Trans. Dependable Secur. Comput
33. Lyu, L., et al.: Towards fair and privacy-preserving federated deep models. IEEE Trans. Parallel Distrib. Syst. **31**(11), 2524–2541 (2020)
34. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282 (2017)
35. McMahan, H.B., Moore, E., Ramage, D., y Arcas, B.A.: Federated learning of deep networks using model averaging. CoRR, arXiv:1602.05629 (2016)
36. McMahan, H.B., Ramage, D., Talwar, K., Zhang, L.: Learning differentially private recurrent language models. In: ICLR (2018)
37. Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: SP, pp. 691–706 (2019)
38. Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning. In: SP, pp. 739–753 (2019)
39. Pan, X., Zhang, M., Ji, S., Yang, M.: Privacy risks of general-purpose language models. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 1314–1331. IEEE (2020)
40. Pan, X., Zhang, M., Wu, D., Xiao, Q., Ji, S., Yang, M.: Justinian's GAAvernor: robust distributed learning with gradient aggregation agent. In: USENIX Security Symposium (2020)
41. Phong, L.T., Aono, Y., Hayashi, T., Wang, L., Moriai, S.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. **13**(5), 1333–1345 (2018)
42. Shafahi, A., et al.: Poison frogs! Targeted clean-label poisoning attacks on neural networks. In: NeurIPS, pp. 6103–6113 (2018)
43. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: SP, pp. 3–18 (2017)
44. Su, L., Xu, J.: Securing distributed machine learning in high dimensions. CoRR, arXiv:1804.10140 (2018)
45. Szegedy, C., et al.: Intriguing properties of neural networks. CoRR, arXiv:1312.6199 (2013)
46. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: KDD, pp. 639–644 (2002)
47. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. (TIST) **10**(2), 1–19 (2019)
48. Yang, Q., Liu, Y., Cheng, Y., Kang, Y., Chen, T., Yu, H.: Federated Learning. Morgan & Claypool Publishers, San Rafael (2019)
49. Yin, D., Chen, Y., Ramchandran, K., Bartlett, P.: Byzantine-robust distributed learning: towards optimal statistical rates. CoRR, arXiv:1803.01498 (2018)
50. Zhao, B., Mopuri, K.R., Bilen, H.: iDLG: improved deep leakage from gradients. CoRR, arXiv:2001.02610 (2020)
51. Zhao, Y., et al.: Local differential privacy based federated learning for Internet of Things. arXiv preprint arXiv:2004.08856 (2020)
52. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: NeurIPS, pp. 14747–14756 (2019)