Towards Mitigating Poisoning Attacks in Federated Learning

Elhattab Fatima, Rania Talbi, Sara Bouchenak, Vlad Nitu

ComPAS'2021 : Parallélisme/ Architecture/ Système / Temps Réel
MILC - Lyon, France, 5-9 juillet 2021

Towards Mitigating Poisoning Attacks in Federated Learning

Fatma-Zohra El Hattab, Rania Talbi, Sara Bouchenak, Vlad Nitu
INSA Lyon – LIRIS, France
{firstname.lastname}@insa-lyon.fr

Abstract
Federated learning is a new machine learning trend that, guided by privacy goals, distributes learning across multiple participants who train the model collaboratively without sharing their data. Nonetheless, it is vulnerable to a variety of attacks such as data and model poisoning. In these attacks, adversaries attempt to inject a backdoor task in the model along with its main task during the training phase. After that, the injected backdoor is exploited at inference-time given a specific trigger. Many state-of-the-art mechanisms that rely on model update auditing have been proposed to mitigate poisoning attacks. We show in this paper that attackers are still capable to evade such detectors by crafting model updates that mimic benign ones. In this paper, we propose ARMOR, a novel mechanism that successfully detects these backdoor attacks in Federated Learning. We describe the design principles of ARMOR based on generative adversarial networks. And we present ARMOR's evaluation results on a real world dataset, which demonstrates that it outperforms its competitors.

1. Introduction

Federated Learning (FL) is a trending framework that allows to carry orchestrated collaborative machine learning without explicitly exchanging training data, thus, improving user data privacy [13]. Due to its interesting guarantees, it was rapidly adopted in many thriving application domains such as next-word prediction [14], speech recognition [4], self-driving cars [9], and many more. Nonetheless, despite its advantages, it has been demonstrated that Federated Learning is highly vulnerable to many client-side attacks since it is user-driven [6]. In this work, we mainly focus on data and model poisoning attacks that target FL robustness [1, 3, 7]. In these attacks, adversaries attempt to inject a backdoor task in the FL model along with its main task. This backdoor assigns an attacker-chosen label to input data with a specific trigger. For instance an attacker can bypass a facial-recognition-based authentication system by assigning to his images a wrong identity-label that is authorized to access the system. Detecting poisoning attacks in Federated Learning is a challenging problem since participants only send model updates to the FL server instead of sending their raw training data. Consequently, the FL server holds less information about user behavior to detect malicious participants. Many state-of-the-art mechanisms have been proposed to mitigate poisoning attacks [2, 8, 10]. Even though these mechanisms have various poisoning detection rules, they all rely on auditing the shape of model updates sent by FL participants to find suspicious ones. In this paper, we show that attackers are still capable of evading such detectors by crafting model updates that mimic benign FL participants' updates. We propose ARMOR, a novel attack detector that is based on GANs, in order to analyze the information that model updates capture about user data, instead of monitoring model updates' geometric shapes.

The rest of this paper is organized as follows. Background and problem statement are introduced in Section 2. ARMOR design principles are described in Section 3. We carry an empirical evaluation comparing ARMOR with state-of-the-art detectors in Section 4, present related work in Section 5, and conclude in Section 6.

2. Background and Motivation

2.1. Background on Federated Learning

Federated Learning (FL) is a new trend for machine learning that provides better user data privacy, by distributing learning across multiple participants who train the model on their local data [13]. For example, multiple participants can jointly train a next-word predictor for keyboards without revealing their information, the data is kept on the participant's device and only the model parameters are transferred to a FL server (a.k.a. aggregator). This allows participants to compute their model updates locally and independently, while maintaining a level of privacy. Federated learning consists of several global rounds, where a typical learning round includes the following sequence : (i) A random subset of participants (i.e., clients) is selected to receive the global model synchronously from the server ; (ii) Each selected client trains the model on its local data and updates it ; (iii) Model updates are sent from the selected clients to the server ; (iv) The server aggregates the received model updates (usually by averaging) to build an improved global model.

2.2. Attacks and Threat Model

In this paper, we are interested in poisoning attacks where an active attacker takes over one or multiple user devices in order to carry targeted poisoning attacks throughout the training process, to provoke misclassification of a subset of particular data samples S to a target class C_t. The attacker crafts his data samples by overlaying a pattern P^* over existing data points and changing their labels to his target class. We consider two types of poisoning attacks proposed by Bagdasaryan et al. in [1]. The first one, being the naive approach (data poisoning attacks), consists in training the model simply on backdoor data as well as correctly labeled data, the attacker can also play on the parameters of the model, such as the learning rate as well as the number of local training epochs, etc. in order to maximize the overfitting on the backdoor data. The second approach relies on model replacement, where the attacker ambitiously attempts to replace the global model with a malicious one by computing the value of the gradient descent that allows poisoning the model. This method guarantees that the attacker's contribution remains effective when aggregated with those of benign participants. Moreover, the results show that this approach allows 'one-shot attack' i.e. the global model has a high accuracy on the backdoor task immediately after its poisoning. Furthermore, [1] demonstrates that model poisoning attacks are much more effective than data poisoning in FL environments, where a single, non-complicit malicious participant aims to make the model misclassify

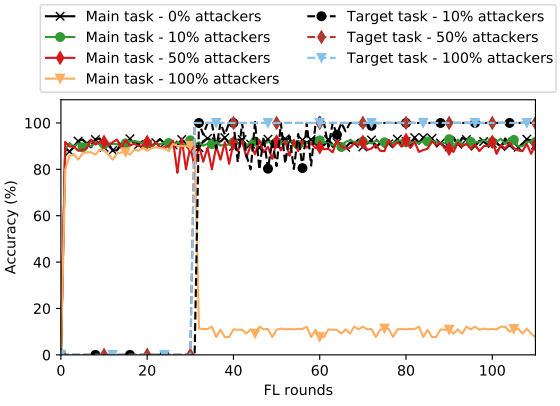


FIGURE 1 – Attack effectiveness under various attack frequencies

a set of chosen inputs with high confidence.

2.3. Problem Illustration

The goal of poisoning attack is to produce a model that achieves high accuracy on both the main task and a backdoor task chosen by the attacker to avoid the detection. Furthermore, the model must maintain high accuracy on the backdoor task for several rounds after the attack, because the attacker is not always selected in the federated learning process. In Figure 1, even with 10% of malicious participants in the system, the attack target task is sustainfully injected in the model without impacting the main task accuracy. However, higher attackers' percentage can deteriorate the model utility.

3. Design Principles of ARMOR

3.1. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are models that try to generate the input distribution of a given model in a way that is as realistic as possible. Roughly speaking, the goal of a GAN is to predict the features of data samples given a label instead of predicting a label based on input data's features. A GAN consists of two models, the first model is called a *Generator* (G), and the second is a *Discriminator* (D). On one hand, the Generator aims to build a model that generates fake inputs of specific targets that are as realistic as possible. On the other hand, the Discriminator has the objective of distinguishing between the fake data of the generative model and the real data. The adversarial training of GANs is done in a way that maximizes both of the aforementioned objectives until a Nash equilibrium is reached.

3.2. Overview of ARMOR

We propose a novel FL data poisoning detection mechanism called *ARMOR* that addresses the threat model described in Section 2.2. Our solution does not make any assumptions neither on the proportion of attackers in the system, nor on their data distributions.

The insight behind ARMOR is as follows. Let T be a backdoor task that aims to misclassify a subset of data samples that belong to a source class C_s and that hold a particular data pattern P^* into a target class C_t. At an FL round i, if the backdoor task T is successfully injected within the model w_i, the class-representatives of the target class C_t that can be generated based on this model, would tend to be confused with the source class C_s, when they are fed to the models of the previous rounds. Based on this intuition, when auditing a model w_i, *ARMOR* monitors the difference between the loss obtained when feeding class representatives to this model w_i and the testing loss obtained when feeding the same data samples to the models of the two previous rounds. If this difference is too high compared to a given threshold, the current model is considered to be corrupted and a rollback to the previous version of the model is necessary. Thus, the detection formula is the following :

$$f(w_i, w_{i-1}, w_{i-2}) = \begin{cases} 1, & \text{if } \frac{L(D,w_i)-L(D,w_{i-1})}{max(L(D,w_i),L(D,w_{i-1}))} > \gamma_1 \text{ and } \frac{L(D,w_i)-L(D,w_{i-2})}{max(L(D,w_i),L(D,w_{i-2}))} > \gamma_2 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

In order to generate class-representatives, *ARMOR* relies on a set of Generative Adversarial Networks (GANs) that are trained on the server-side based on users' model updates. The total number of these GAN models is equal to the number of target classes of the FL model, where each GAN is trained to generate data samples of a given class at a round i. In ARMOR, each

GAN network GAN_{ki} used to build representatives of a class C_k at an FL round i consists of a *generator* and a *discriminator*. The generator outputs data points that look like data of a given label. The discriminator has an identical structure as the federated model except that it is augmented with an additional class that represents fake data samples. The generator is trained while maximizing the number of data points that bypass the discriminator while the latter has the opposite objective. The adversarial training of the two models is carried through multiple epochs, where each epoch consists of three phases. First, the discriminator is updated based on real data samples. Since the server does not hold real data points, the latter is gradually updated using the local user updates. After that, the discriminator is updated using fake data samples obtained from the generator, and last, the generator is updated based on the output of the discriminator when observing fake data. The process is repeated until the generator outputs data samples that are classified to the target class with high accuracy. The resulting class representatives are then fed to the current model w_i and the last two models from previous rounds w_{i-1}, and w_{i-2}. The testing loss is computed for each one these model versions and then used to detect poisoning via formula 1. If a poisoning is detected a rollback to the previous model version is done. The detailed detection process implemented in ARMOR is provided in Algorithm 3.2.

Algorithm 1 ARMOR Poisoning Detection

Inputs : Model at current round w_i, models at previous rounds w_{i-1}, and w_{i-2}
At each iteration i **do :**
for $k \in 1..K$ **do**
  Consider $GAN_{ik} = (G_{ik}, D_{ik})$
  for $e \in 1..Epochs$ **do**
    $D_{ik} \leftarrow AverageModels(w_i * \frac{1}{epochs}, D_{ik})$
    Generate a random noise vector $X^{(t)} \leftarrow Random()$
    Feed the random noise $X^{(t)}$ to the generator to get fake data $F^{(t)} \leftarrow G_{ik}(X^{(t)})$
    Feed the fake data $F^{(t)}$ to the discriminator $p^{(t)} \leftarrow D_{ik}(F^{(t)})$
    Update D_{ik} by computing $L(p^{(t)}, C_{fake})$
    Update G_{ik} by computing $L(p^{(t)}, k)$
    if $test\_accuracy(F^{(t)}, w_i) > \alpha$ **then** : break
  end for
  Output the class-representatives $Z_k \leftarrow F^{(t)}$
  Feed the class-representatives to w_i, w_{i-1}, and w_{i-2}
  $y_i \leftarrow w_i(Z_k)$, $y_{i-1} \leftarrow w_{i-1}(Z_k)$, $y_{i-2} \leftarrow w_{i-2}(Z_k)$
  Compute the testing loss for w_i, w_{i-1}, and w_{i-2}
  $L_i \leftarrow L(y_i, k)$, $L_{i-1} \leftarrow L(y_{i-1}, k)$, $L_{i-2} \leftarrow L(y_{i-2}, k)$
  if $f(w_i, w_{i-1}, w_{i-2}) = 1$ **then** :
  Rollback to the previous model version $w_i = w_{i-1}$
  break
end for

4. Experimental Evaluation

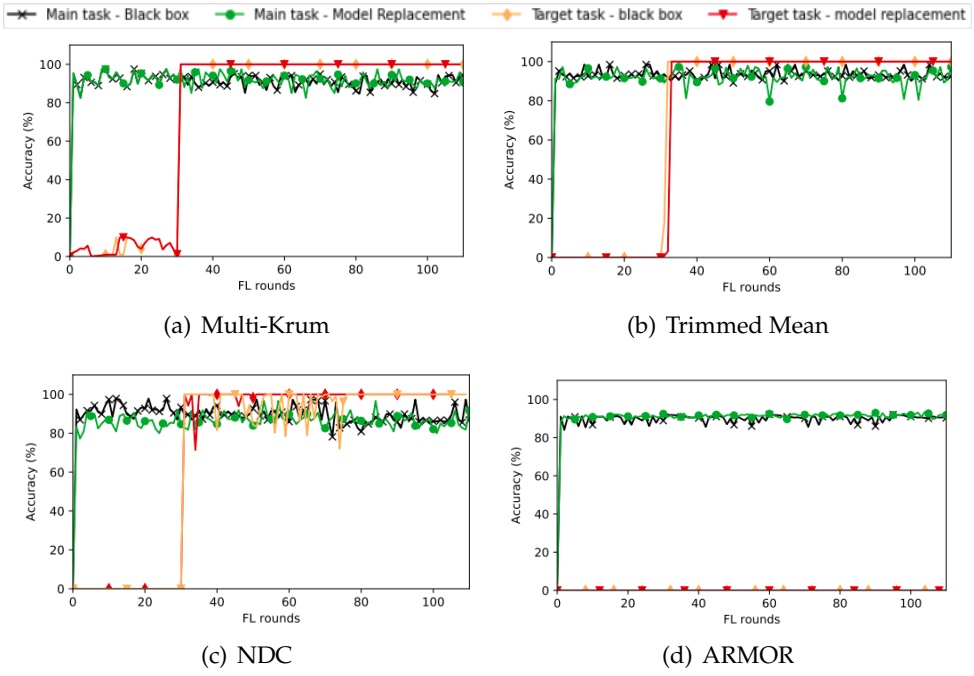
4.1. Experimental Setup

Our experiments are conducted using the real-world Fashion-MNIST dataset, which contains 50,000 training images and 10,000 test images [12]. We trained a model with 10 clients selected each round. We employ a 4-layer Convolutional Neural Network and simulate a non-IID data

distribution using the Dirichlet distribution [5]. In each learning epoch, the clients train the models with 10 local epochs, and with a learning rate $\lambda = 0.1$. To evaluate our detector, we have implemented an attack that consists in modifying the classification for the images that contain an artificial pattern, added on the images in the attacker's dataset. Furthermore, we trained the model with the projected gradient descent (PGD) [11] so that the attacker's model does not diverge much from the global model in each epoch.

4.2. Evaluation Results

To test the efficiency of our detector, we analyzed three state-of-the-art detectors MultiKrum [2], Trimmed Mean [7] and Norm Difference Clipping (NDC) [10]. For each detector, we examine the accuracy of the main task, and the success rate of the attack with different poisoning methods (i) blackbox (data poisoning) where the attacker only alters the data in order to insert a backdoor, but respects the FL optimization algorithm, (ii) whitebox (model poisoning) : the attacker disobeys the protocol and relies on model replacement to substitute the global model with their local model.



(a) Multi-Krum (b) Trimmed Mean

(c) NDC (d) ARMOR

FIGURE 2 – Attack effectiveness under various state-of-the-art defenses

Figure 2 shows the accuracy of the main task and the success rate of the attack obtained with Multi-Krum, Trimmed Mean, NDC and ARMOR detectors respectively. Our experiments are performed over more than 100 rounds of training and we assume that for the first 30 rounds no attack is introduced. Further, an attack is introduced with attackers that take over 10% of participants. In order to evaluate the detector in an aggressive scenario, the attack occurs every epoch. We can see on Figures 2(a), 2(b) and 2(c) that from the introduction of the attack, it reaches a success rate of 100% that is kept throughout the learning rounds with the state-of-the-art detectors. In fact, MultiKrum and Trimmed Mean use the hypothesis that malicious updates are far from benign updates, but the PGD-based attack allows to reduce the deviation between the attacker's model and the global model, so the attacker is not detected. Regarding NDC, the attack is successfull in both the black-box and white-box scenarios but its accuracy is slightly noisy because the updates that exceed the norm threshold are clipped. In Figure 2(d),

we can observe that ARMOR is able to detect the attack even if it occurs in every epoch. Due to label flipping, the features of class representatives are modified and therefore, a loss difference between the model at epoch t and epoch $t-1$ helps to detect the attack.

5. Related Work

Detecting attacks is generally much more challenging than attacking the FL framework. Many of the protection mechanisms are coming from the domain of distributed ML and rely on robust statistic mechanisms such as trimmed mean or geometric median. One of the most prominent and popular protection mechanisms of this kind is Krum [2] which relies on the robust property of the median to measure the central tendency of the gradient. At the end of each epoch and for every received model update, the FL server will sum the distance to its $n-f-2$ closest neighbours. Finally, the server will compute a gradient step with the update that minimize the above computed sum. In the geometric representation of the model updates, this is the vector closest to the barycenter. The authors also presented an improved variant (called Multi-Krum) which interpolates between Krum and averaging, thereby allowing to mix the resilience properties of Krum with the convergence speed of averaging. Pillutla et al. [8] introduces RFA, an algorithm which aggregates the local models by computing a weighted geometric median using the smoothed Weiszfeld's algorithm. However, these mechanisms based on geometric median rely on a central assumption which is often not true in FL setups. They assume that all training participants have similar learning objectives therefore their model updates will point to similar directions. Due to its relatively different objective, the vector sent by an attacker will differentiate from the other honest participants and can be filtered out. However, in FL, the data is non independent and identically distributed (non-IID) among the paritcipants so their update vectors will be more scattered in space. Sun et al. [10] relies on the assumption that attackers will send model updates with larger norms and therefore introduces a protection mechanism based on norm-clipping. The model updates with large norms will be reduced so that their magnitude will become comparable with all other model updates. In the same paper, the authors introduced another model protection mechanisms based on differential privacy. The FL server perturbs all model updates received from the FL workers with Gaussian noise. Therefore, the attack objective of a malicious user will be slightly diverged, but with the cost of also degrading the quality of honest updates. ARMOR relies on a more elaborate detection mechanisms so the malicious model updates can be individually identified and filtered out. Unlike the detection mechanisms presented above, *ARMOR* does not rely on auditing the shape of model updates, but instead focuses on their informational essence and the impact that they have on the trained model. As we have shown in Section 4, ARMOR overpasses all state-of-the-art detection mechanisms.

6. Conclusion and Future Work

In this paper we propose ARMOR a GAN-based detection mechanism that mitigates targeted data and model poisoning attacks in Federated Learning. ARMOR analyzes the information captured by model updates about user data, instead of monitoring their geometric shapes like other state-of-the-art mechanisms. Therefore, we have demonstrated the effectiveness of our detection mechanism against various poisoning attack scenarios that state-of-the-art detection mechanisms fail to detect. In future work, we intend to extend our evaluation by considering other machine learning tasks such text analysis. We are also exploring other improved poisoning detection strategies that determine the exact attack source as well as reduce the computational overhead induced by our solution.

Bibliographie

1. Bagdasaryan (E.), Veit (A.), Hua (Y.), Estrin (D.) et Shmatikov (V.). – How to backdoor federated learning, 2019.
2. Blanchard (P.), El Mhamdi (E. M.), Guerraoui (R.) et Stainer (J.). – Machine learning with adversaries : Byzantine tolerant gradient descent. – In Guyon (I.), Luxburg (U. V.), Bengio (S.), Wallach (H.), Fergus (R.), Vishwanathan (S.) et Garnett (R.) (édité par), *Advances in Neural Information Processing Systems*volume 30. Curran Associates, Inc., 2017.
3. Fung (C.), Yoon (C. J.) et Beschastnikh (I.). – Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv :1808.04866*, 2018.
4. Guliani (D.), Beaufays (F.) et Motta (G.). – Training speech recognition models with federated learning : A quality/cost framework. *arXiv preprint arXiv :2010.15965*, 2020.
5. Hsu (T.-M. H.), Qi (H.) et Brown (M.). – Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv :1909.06335*, 2019.
6. Lyu (L.), Yu (H.) et Yang (Q.). – Threats to federated learning : A survey. *arXiv preprint arXiv :2003.02133*, 2020.
7. Mhamdi (E. M. E.), Guerraoui (R.) et Rouault (S.). – The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv :1802.07927*, 2018.
8. Pillutla (K.), Kakade (S. M.) et Harchaoui (Z.). – Robust aggregation for federated learning, 2019.
9. Pokhrel (S. R.) et Choi (J.). – A decentralized federated learning approach for connected autonomous vehicles. – In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6. IEEE, 2020.
10. Sun (Z.), Kairouz (P.), Suresh (A. T.) et McMahan (H. B.). – Can you really backdoor federated learning ? *CoRR*, vol. abs/1911.07963, 2019.
11. Wang (H.), Sreenivasan (K.), Rajput (S.), Vishwakarma (H.), Agarwal (S.), yong Sohn (J.), Lee (K.) et Papailiopoulos (D.). – Attack of the tails : Yes, you really can backdoor federated learning, 2020.
12. Xiao (H.), Rasul (K.) et Vollgraf (R.). – Fashion-mnist : a novel image dataset for benchmarking machine learning algorithms. *CoRR*, vol. abs/1708.07747, 2017.
13. Yang (Q.), Liu (Y.), Chen (T.) et Tong (Y.). – Federated machine learning : Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, n2, 2019, p. 12.
14. Yang (T.), Andrew (G.), Eichner (H.), Sun (H.), Li (W.), Kong (N.), Ramage (D.) et Beaufays (F.). – Applied federated learning : Improving google keyboard query suggestions. *arXiv preprint arXiv :1812.02903*, 2018.