LEARNING TO BACKDOOR FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In a federated learning (FL) system, malicious participants can easily embed backdoors into the aggregated model while maintaining the model's performance on the main task. To this end, various defenses, including training stage aggregation-based defenses and post-training mitigation defenses, have been proposed recently. While these defenses obtain reasonable performance against existing backdoor attacks, which are mainly heuristics based, we show that they are insufficient in the face of more advanced attacks. In particular, we propose a general reinforcement learning-based backdoor attack framework where the attacker first trains a (non-myopic) attack policy using a simulator built upon its local data and common knowledge on the FL system, which is then applied during actual FL training. Our attack framework is both adaptive and flexible and achieves strong attack performance and durability even under state-of-the-art defenses.

1 INTRODUCTION

A backdoor attack against a deep learning model is one where a backdoor is embedded into the model at the training stage and is triggered at the test stage only for targeted data samples. Such an attack is practical and difficult to defend due to its stealthiness and flexibility. The problem becomes more interesting and challenging (for both the attacker and the defender) in a distributed learning system such as federated learning (FL), where devices collaboratively train a model without sharing their local data (McMahan et al., 2017). In this setting, a malicious insider can only control its local model and has limited influence on the global model, while the server only has access to the models shared by the devices but not their local data. Despite these difficulties, Bagdasaryan et al. (2020) shows that when there is no defense, even a single malicious insider can inject a backdoor into the global model while maintaining its performance on the main task by first training a backdoored model using local data and then poisoning the global model through model replacement to amplify the backdoor effect. The stealth of the poisoned updates can be further improved by distributing triggers across multiple cooperative malicious devices (Xie et al., 2019) and by considering edge-case backdoors (Wang et al., 2020), making it possible to bypass certain training stage defenses such as norm-bounding (Sun et al., 2019) and weak differentially private (Geyer et al., 2017) defenses.

An important limitation of existing backdoor attacks against federated learning is that they often target specific types of defenses or require a relatively large number of malicious devices to be effective. Further, none of them can compromise post-training defenses (Wu et al., 2020; Nguyen et al., 2021; Rieger et al., 2022) as we show in our experiments. The main reason is that these attacks are myopic and largely ignore the impact of potential defenses on (long-term) attack performance. One exception is Anticipate (Wen et al., 2022), which estimates the impact of malicious updates by simulating the system's response to attacks in the next few steps, assuming federated averaging is used by the server. However, due to the simplifications made, it fails under defenses such as Krum and Median.

The limited success of existing backdoor attacks might give a false sense of robustness of FL systems against backdoors. In this work, we show that there exists a strong backdoor attack that can compromise state-of-the-art defenses with a relatively small number of malicious devices. The main idea is to formulate the backdoor attack problem as a Markov decision process (MDP) and utilize the local data of malicious devices and common knowledge to build a simulator of the MDP, that is, a "world model" of the real FL environment under attacks. The reward function of the MDP takes into consideration both the main task and the backdoor task. The set of malicious devices first trains an attack policy using deep reinforcement learning (RL) in the simulated environment, which is then applied in the real FL training process. Our RL-based attack framework can be applied to

both data poisoning and model poisoning. We focus on the latter in this work due to its flexibility. To reduce the action space, we design a Double Whammy Backdoor Attack (DWBA) method. Rather than crafting a backdoored model from scratch, we first adopt a two-task learning approach to updating the local model in each FL round and then adjust its weights using ideas from Zhang et al. (2022). RL is used to optimize the hyperparameters of both steps alternatively. Despite action space compression and the gap between the simulated and real environments, our RL-based attack achieves high accuracy for both the main task and the backdoor task and better backdoor durability under state-of-the-art training stage aggregation-based defenses and post-training mitigation defenses. Our results indicate the importance of developing more adaptive and stronger defenses against backdoor attacks.

2 Reinforcement Learning Backdoor Attack Framework

Federated Learning. A federated learning system consists of a server and K clients where each client has a local dataset. Let $D_k = \{(x_k^i, y_k^i)_{i=1}^{n_k}\}$ denote the local dataset of the k-th client and $n_k = |D_k|$ the size of the dataset. The empirical loss of a model w for the k-th client is defined as $F_k(w) = f(w, D_k) := \frac{1}{n_k}\sum_{i=1}^{n_k} \ell(w, (x_k^i, y_k^i))$, where $\ell(\cdot,\cdot)$ is the loss function. Let $U = \{D_1, D_2, \ldots, D_K\}$ denote the collection of all local datasets. The goal of federated learning is to find a model w that minimizes the average loss across all the devices: $\min_w F(w, U) := \frac{1}{K}\sum_{k=1}^{K} f(w, D_k)$.

In each federated training round t over T total rounds, the server distributes a global model w_g^t to a set of randomly selected clients $\mathcal{S}^t$ and let $\kappa = |\mathcal{S}^t|/K$ denotes the subsampling rate. Each selected client k then performs E local iterations of stochastic gradient descent to update w_g^t to obtain a local model w_k^t with its own data and sends the model update $g_k^t = w_k^t - w_g^t$ back to the server. The server then aggregates the set of model updates $\{g_k^t\}_{k\in\mathcal{S}_t}$ from the selected clients using a certain aggregation rule $Aggr$. The global model is then updated as $w_g^{t+1} = w_g^t - Aggr(\{g_k^t\}_{k\in\mathcal{S}^t})$, which is then distributed to the selected clients in the $(t+1)$-th round of training. Various $Aggr$ rules have been proposed in the literature, including simple federated averaging as well as robust aggregation rules such as norm-bounding, Krum, and Median. Further, the server may perform a post-training defense $h(\cdot)$ such as Neuron Clipping (Wang et al., 2022) and Pruning (Wu et al., 2020) at round T on the global model to obtain $\widehat{w}_g^T = h(w_g^T)$.

Threat Model. Without loss of generality, we assume that among the K clients, the first M of them are malicious. The set of malicious devices is assumed to be fully cooperative. In each FL round, each malicious device k (if selected by the server) sends a crafted local update $\widetilde{g}_k^t$ to the server. The goal of malicious devices is to inject a backdoor into the global model so that the model misclassifies any test inputs with a chosen pattern (called a backdoor trigger) embedded to a target label chosen by the attackers ("backdoor task") while maintaining good performance on clean test inputs ("main task"). In this work, we assume that the backdoor trigger and the target label are pre-chosen and focus on how to best embed the backdoor into the global model (i.e., model poisoning). To this end, each malicious device $k \in [M]$ first builds a crafted dataset D_k' that is derived from its clean dataset D_k by adding the trigger to a subset of D_k. Let $\rho_k = |D_k'|/|D_k|$ denote the poison ratio. Let $U' = (D_1', ..., D_M')$ denote the collection of poisoned datasets. Then the main task is $\min_w F(w, U)$ as defined above, while the backdoor task is $\min_w F(w, U') := \frac{1}{M}\sum_{k=1}^{M} f(w, D_k')$. Similarly to Bagdasaryan et al. (2020) and Xie et al. (2019), we define the attacker's goal as $\min_w F'(w) := \lambda F(w, U) + (1-\lambda) F(w, U')$ where $\lambda \in [0, 1]$ is a hyperparameter for making a tradeoff between the two tasks.

2.1 Reinforcement Learning based Backdoor Attack Framework

Unlike previous work that solves the attacker's problem for a single FL round, we consider a backdoor attack as a sequential decision-making problem and formulate it as a Markov decision process (MDP), which can be represented as a tuple (S, A, T, R, γ) where

- S is the state space. The state at round t is defined as $s^t := (w_g^t, \mathcal{A}^t)$, where w_g^t is the global model parameters and $\mathcal{A}^t$ is the set of malicious devices sampled by the server, both at round t.
- A is the action space. Let $a^t := \{\widetilde{g}_k^t\}_{k=1}^{M}$ denote the joint action of malicious devices at round t. Note that a malicious device not sampled at round t does not send any information to the server; hence its action has no effect on the model update.

- $T : S \times A \rightarrow \mathcal{P}(S)$ is the state transition function, which is jointly determined by the number of benign devices and that of malicious devices, their local datasets and training methods, subsampling rate, and the defense mechanism the server applies.
- $R : S \times A \rightarrow \mathbb{R}_{\leq 0}$ is the reward function. Given state s^t and action a^t, we define the expected reward at round t as $r^t := -\mathbb{E}[F'(\widehat{w}_g^{t+1})] = -\mathbb{E}[F'(h(w_g^t - Aggr(\{\widetilde{g}_i^t\} \cup \{g_j^t\}_{i,j \in \mathcal{S}^t, i \in [M], j \notin [M]})))]$ if $\mathcal{A}^t \neq \emptyset$, and $r^t := 0$ otherwise.
- $\gamma \in (0, 1)$ is the discount factor for future rewards.

The attacker's objective is to find a stationary policy $\pi : S \rightarrow A$ that maximizes the expected total reward over T FL rounds, which is equivalent to minimize $\sum_{t=0}^{T-1} \gamma^t \mathbb{E}[F'(\widehat{w}_g^{t+1})]$.

Simulated Environment. To derive the optimal backdoor attack policy, the problem then boils down to solving the MDP defined above. One approach is to apply online reinforcement learning, where the set of malicious devices collaboratively update their attack polices during the interaction with the real FL environment. Given the limited amount of attack opportunities and feedback available, however, this approach is unlikely to work in practice. We instead adopt an offline approach in this work. In particular, we consider the commonly assumed white-box setting where the attacker has prior knowledge about the FL environment including the server's defense mechanism, the local training method, the number of devices, and the subsampling rate. However, they do not have access to the local datasets of benign devices. Instead, they use their own data to approximate the local data distributions of other devices. With this information, the attacker can approximate both the transition function and the reward function of the MDP to build a simulator for it. This can be done before the actual FL training starts or with limited amount of online interaction. With the simulator built, the set of malicious devices can adopt a deep reinforcement learning algorithm such as TD3 (Fujimoto et al., 2018) or PPO (Schulman et al., 2017) to solve the MDP to train an attack policy, which is then applied during the actual FL training.

2.2 Double Whammy Backdoor Attack

For the scalability of our attack against large FL models, we propose to only include in the state the model parameters of the last two hidden layers of w_g^t and the number of attackers sampled at round t (instead of the entire set of attackers). We further restrict all the malicious devices to take the same action. To further compress the action space, we consider the following two-step approach for crafting gradients and use reinforcement learning to optimize the hyperparameters of both steps.

Local Search. First, consider a local objective function $\tilde{F}'(w) := \lambda F(w, \{D_k\}_{k \in [M]}) + (1 - \lambda) F(w, \{D'_k\}_{k \in [M]})$, which is an approximation of $F'(w)$ using attacker's local data (including both clean data and poisoned data pooled together from all malicious devices). At each FL round t, the attacker first generates a new model $\tilde{w}^t$ by updating the current global model w_g^t via E^t steps of stochastic gradient descent with respect to $F'(w)$ under earning rate η^t, batch size B^t, and poison ratio ρ^t (see Algorithm 1 in the Appendix). This step is similar to the method in Bagdasaryan et al. (2020) and Xie et al. (2019). Let $\tilde{g}^t = w_g^t - \tilde{w}^t$ denote the model update.

Model Crafting. For each layer of $\tilde{g}^t$, the attacker modifies a fraction α^t of parameters that differ the most from the corresponding parameters of g^t, the model update calculated by using attacker's clean data only. In particular, we let $\tilde{g}^t[i] = \tilde{g}^t[i] - \beta^t(\tilde{g}^t[i] - g^t[i])$ for each selected coordinate i where β^t is a one-dimension scaling factor. This $\tilde{g}^t$ is what each sampled malicious device will send to the server at round t. This step is similar to Zhang et al. (2022) in spirit. To avoid detection, a small amount of random noise can be added to the crafted gradient at each malicious device.

The concrete approach used in each of the above steps can be replaced by other algorithms. The key novelty of our approach is that instead of using fixed and hand-crafted hyperparameters, i.e., $a_1^t := (\rho^t, B^t, E^t, \eta^t)$ in the first step and $a_2^t := (\alpha^t, \beta^t)$ in the second step, as in existing approaches, we use RL to optimize them. Rather than searching for all the parameters together, the two sub-actions are optimized alternatively. The details are given in the Appendix.

3 Experiment

In this section, we compare our RL-based attack with several state-of-the-art backdoor attacks against federated learning, including BFL (Bagdasaryan et al., 2020), DBA (Xie et al., 2019), PGD attack (Wang et al., 2020), Neurotoxin (Zhang et al., 2022), and Anticipate (Wen et al., 2022), under common training stage aggregation-based defenses including FedAvg (McMahan et al., 2017),

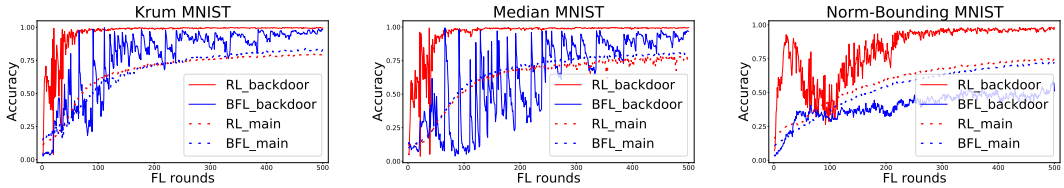

Figure 1: Global model accuracy for the backdoor and main tasks during FL training, under BFL and RL attack, and Krum, Median, and norm-bounding defenses. All parameters are set as default.

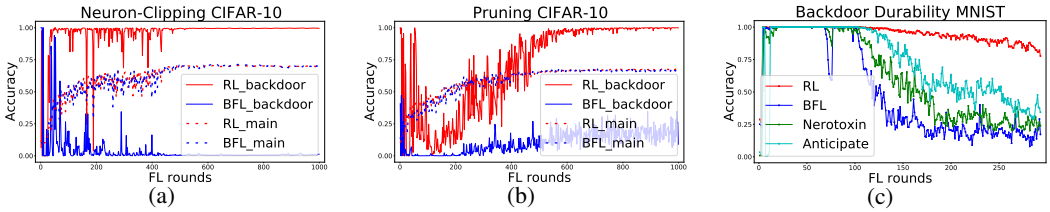

Figure 2: (a)-(b) Global model accuracy for the backdoor and main tasks during FL training, under BFL and RL attack, and Neuron Clipping and Pruning defenses. (c) Backdoor durability of the RL attack, BFL, Neurotoxin, and Anticipate. Attacks are applied during the first 100 FL rounds only.

Krum (Blanchard et al., 2017), Median (Yin et al., 2018), and norm-bounding (Sun et al., 2019), and/or post-training mitigation defenses including Neuron Clipping (Wang et al., 2022) and Pruning (Wu et al., 2020). Due to the space limit, we moved a detailed description of the experiment setup and additional results to the Appendix.

Training Stage Defenses. Figure 1 shows how the backdoor accuracy and the main-task accuracy of the global model (with respect to the test data) vary over FL training, under BFL and our RL attack when the server implements Krum, Median, and norm-bounding defenses, respectively. The default threshold for the norm-bounding defense is 0.05. Results for DBA and PGD under training stage defenses can be found in the Appendix. For both Krum and Median, our RL attack reaches $\sim 100\%$ backdoor accuracy within 100 FL rounds and remains stable afterwards, while the backdoor performance of BFL fluctuates during FL training due to subsampling and never reaches the same level of high accuracy as the RL attack does. Further, our RL attack maintains similar accuracy for the main task as the baselines. For the norm-bounding defense, our RL attack outperforms BFL for both the backdoor and main tasks. In particular, for the backdoor task, the RL attack again reaches $\sim 100\%$ accuracy after 220 FL rounds, while the backdoor accuracy of BFL slowly grows and never goes beyond 50% during FL training.

Post-Training Defenses. Figures 2(a) and 2(b) show the test accuracy of the global model with respect to the backdoor and main tasks under BFL and our RL attack when the server uses FedAvg as the training stage aggregation rule and Neuron Clipping or Pruning as the post-training defense. We envision that a post-training defense can be applied to an intermediate global model during FL training, although its performance can be unstable before the main task is close to convergence. Thus, we include it in computing the reward function during RL training and in evaluating the test accuracy of an intermediate global model. We observe that both defense methods can effectively decrease the backdoor accuracy for BFL. However, our RL attack bypasses both of them by incorporating their influence into its reward function.

Durability. In Figure 2(c), we compare the backdoor durability of our RL attack, BFL, Neurotoxin, and Anticipate, with FedAvg as the aggregation rule and no post-training defense applied. Attacks are applied during the first 100 FL rounds only. We observe that all attack methods reach high backdoor accuracy ($\sim 100\%$) while the attack happens. After the first 100 FL, our RL attack shows the slowest accuracy decay among all attacks. At round 300, our RL attack still maintains 80% backdoor accuracy, while other attacks' backdoor accuracy is all below 40%. The reason behind the high durability of our attack is that it optimizes for a long-term attack objective by simulating the future behavior of the FL system under attack.

REFERENCES

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *Artificial Intelligence and Statistics (AISTATS)*, pp. 2938–2948. PMLR, 2020.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning (ICML)*, pp. 1587–1596. PMLR, 2018.

Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 770–778, 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics (AISTATS)*, pp. 1273–1282. PMLR, 2017.

Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, et al. Flame: Taming backdoors in federated learning. *Cryptology ePrint Archive*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems (NeurIPS)*, 32, 2019.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1):12348–12355, 2021.

Phillip Rieger, Thien Duc Nguyen, Markus Miettinen, and Ahmad-Reza Sadeghi. Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection. *arXiv preprint arXiv:2201.00763*, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? In *NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

Hang Wang, Zhen Xiang, David J Miller, and George Kesidis. Universal post-training backdoor detection. *arXiv preprint arXiv:2205.06900*, 2022.

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In *Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 16070–16084, 2020.

Yuxin Wen, Jonas Geiping, Liam Fowl, Hossein Souri, Rama Chellappa, Micah Goldblum, and Tom Goldstein. Thinking two moves ahead: Anticipating other users improves backdoor attacks in federated learning. In *ICML Frontiers Workshop (AdvML)*, 2022.

Chen Wu, Xian Yang, Sencun Zhu, and Prasenjit Mitra. Mitigating backdoor attacks in federated learning. *arXiv preprint arXiv:2011.01767*, 2020.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning (ICML)*, pp. 5650–5659. PMLR, 2018.

Zhengming Zhang, Ashwinee Panda, Linyue Song, Yaoqing Yang, Michael Mahoney, Prateek Mittal, Ramchandran Kannan, and Joseph Gonzalez. Neurotoxin: Durable backdoors in federated learning. In *International Conference on Machine Learning (ICML)*, pp. 26429–26446. PMLR, 2022.

APPENDIX

A THE BACKDOOR ATTACK ALGORITHM

Algorithm 1 Federated Learning with Double Whammy Backdoor Attack

Server Input: Initial weight w_g^0, set of all clients $[K]$, all clients' local datasets $\{D_1, D_2, \cdots, D_K\}$, subsampling rate κ, aggregation rule $Aggr$, post-training defense $h(\cdot)$
Client Input: Local minibatch size B, local step size η, local iteration number E, number of global training steps T, local search policy π_1, model crafting policy π_2
Server Output: $\widehat{w}_g^T$

Server executes:
  for $t = 0$ to $T-1$ **do**
    $\mathcal{S}^t \leftarrow$ randomly sample $\kappa \cdot K$ clients
    Broadcast w_g^t to all clients in $\mathcal{S}^t$
    Wait for all model updates $\{g_k^t\}_{k\in\mathcal{S}^t}$
    Update $w_g^{t+1} \leftarrow w_g^t - Aggr(\{g_k^t\}_{k\in\mathcal{S}^t})$
  end for
  $\widehat{w}_g^T \leftarrow h(w_g^T)$

BenignUpdate(k, w):
  $w_0 \leftarrow w$
  for local epoch $e = 1$ to E **do**
    Sample a minibatch b of size B from D_k
    $w \leftarrow w - \eta \frac{1}{B} \sum_{(x,y)\in b} \nabla_w \ell(w, (x, y))$
  end for
  $g \leftarrow w_0 - w$
  Send g to server

DoubleWhammyUpdate(k, w):
  Malicious devices communicate to get $|\mathcal{A}|$
  $\rho, B', E', \eta' \leftarrow \pi_1(w, |\mathcal{A}|)$
  $\alpha, \beta \leftarrow \pi_2(w, |\mathcal{A}|)$
  $D'_k \leftarrow$ randomly poison local dataset D_k with poison ratio ρ
  $w_0 \leftarrow w$ and $w' \leftarrow w$
  for local epoch $e = 1$ to E' **do**
    Sample a minibatch b' of size B' from D'_k
    $w' \leftarrow w' - \eta' \frac{1}{B'} \sum_{(x,y)\in b'} \nabla_w \ell(w, (x, y))$
  end for
  $\tilde{g} \leftarrow w_0 - w'$
  for local epoch $e = 1$ to E **do**
    Sample a minibatch b of size B from D_k
    $w \leftarrow w - \eta \frac{1}{B} \sum_{(x,y)\in b} \nabla_w \ell(w, (x, y))$
  end for
  $g \leftarrow w_0 - w$
  for each layer l of $\tilde{g}$ **in parallel do**
    Select top-$\alpha\%$ coordinates of $|\tilde{g}[l] - g[l]|$
  end for
  for each coordinate i of all selected coordinates **in parallel do**
    $\tilde{g}[i] \leftarrow \tilde{g}[i] - \beta(\tilde{g}[i] - g[i])$
  end for
  Send $\tilde{g}$ to server

B EXPERIMENT SETUP

Datasets. We consider two datasets: MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky, 2009), and $i.i.d.$ local data distributions, where we randomly split each dataset into K groups, each with the same number of training samples. MNIST includes 60,000 training examples and 10, 000

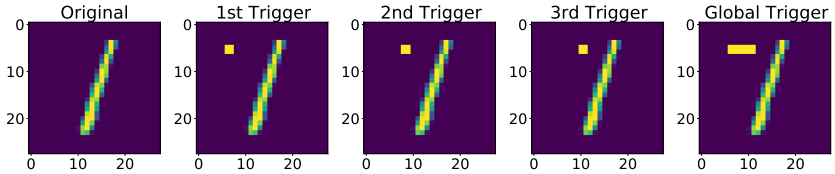

Figure 3: MNIST backdoor trigger patterns. The global trigger is considered the default poison pattern and is used for backdoor accuracy evaluation. The sub-triggers are used by DBA only.

testing examples, where each example is a 28×28 grayscale image, associated with a label from 10 classes. CIFAR-10 consists of 60,000 color images in 10 classes of which there are 50, 000 training examples and 10,000 testing examples. We consider the trigger patterns shown in Figure 3 and Figure 4, where the goal is to misclassify digit 1 to digit 7 for MNIST and airplane class to truck class for CIFAR-10. The default poison ratio is 0.5 in both cases. We consider both a global trigger and a set of sub-triggers for each dataset as shown in the figures, similar to DBA Xie et al. (2019). The global trigger is used as the default poison trigger pattern and the pattern for evaluating the test accuracy of all backdoor attacks, while the sub-triggers are only used by DBA.

Federated Learning Setting. We use the following default parameters for the FL environment: local minibatch size = 128, local iteration number = 1, learning rate = 0.05, number of workers = 100, number of attackers = 5, subsampling rate = 10%, and the number of FL training rounds = 500 (resp. 1000) for MNIST (resp. CIFAR-10). For MNIST, we train a neural network classifier consisting of two 5×5 convolutional filter layers with ReLU activations followed by two fully connected layers and softmax output. For CIFAR-10, we use the ResNet-18 model (He et al., 2016).We implement the FL model with PyTorch (Paszke et al., 2019) and run all the experiments on the same 2.30GHz Linux machine with 16GB NVIDIA Tesla P100 GPU. We use the cross-entropy loss as the default loss function and stochastic gradient descent (SGD) as the default optimizer. For all the experiments, we fix the random seeds of subsampling for fair comparisons.

Baselines. We compare our RL-based attack (RL) with the state-of-the-art backdoor FL attack methods: BFL (Bagdasaryan et al., 2020) without model replacement, DBA (Xie et al., 2019) where each selected attacker randomly chooses a sub-trigger as shown in Figures 3 and 4, PGD attack (Wang et al., 2020) with a projection norm of 0.05, Neurotoxin (Zhang et al., 2022) with top-100 ($\sim 1\%$) parameters for masking at each layer. and Anticipate (Wen et al., 2022) with 5 look-ahead steps. For training stage defenses, we consider four aggregation-based methods: FedAvg (McMahan et al., 2017), Krum (Blanchard et al., 2017), Median (Yin et al., 2018), and norm-bounding (Sun et al., 2019) (with a norm bound of 0.02). For post-training defenses, we consider the backdoor mitigation methods in Wang et al. (2022) (Neuron Clipping) and Wu et al. (2020) (Pruning). We use the original clipping thresholds in Wang et al. (2022) and set the default pruning number to 256.

Reinforcement Learning Setting. In our RL-based attack, since both the action space and state space are continuous, we choose the state-of-the-art Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) algorithm to alternatively train the local search policy for 10,000 steps and the model crafting policy for 10,000 steps in each iteration, with 80,000 total training steps (i.e., the number of training iterations is 4). We implement our simulated environment with OpenAI Gym (Brockman et al., 2016) and adopt OpenAI Stable Baseline3 (Raffin et al., 2021) to implement TD3. The RL training parameters are described as follows: the number of FL rounds = 500 rounds, policy learning rate = 0.001, the policy model is MultiInput Policy, batch size = 256, and γ = 0.99 for updating the target networks. The default $\lambda = 0.5$ when calculating the rewards. For Median (resp. Krum), we set $\lambda = 0.4$ (resp. $\lambda = 0.375$).

As described in Section 2.2, we split the attackers' action into two sub-actions a_1^t and a_2^t corresponding to the local search step and the model crafting step, respectively. Let A_1 and A_2 denote the action spaces in the two steps and define the local search policy and the model crafting policy as $\pi_1 : S \to A_1$ and $\pi_2 : S \to A_2$, respectively. Instead of training $\pi = (\pi_1, \pi_2)$ simultaneously, we implement an alternative training structure in the simulated environment, where we alternatively train π_1 and π_2 while fixing the other policy. Figure 7(b) shows the advantage of this approach compared with simultaneous training.

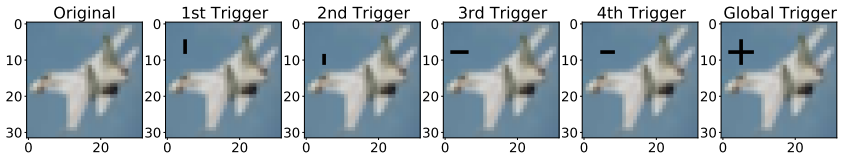

Figure 4: CIFAR-10 backdoor trigger patterns.The global trigger is considered the default poison pattern and is used for backdoor accuracy evaluation. The sub-triggers are used by DBA only.

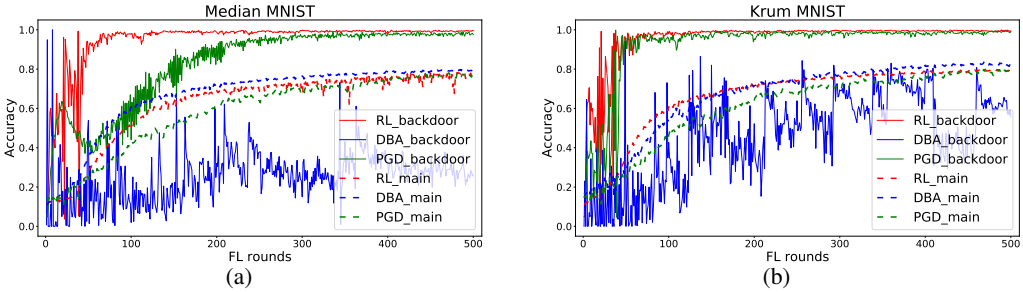

Figure 5: A comparison of the global model accuracy for the backdoor task and the main task during FL training, under RL attack, DBA, and PGD attack with Krum and Median defenses. All parameters are set as default.

C More Experiment Results

C.1 Attack Performance

More Baseline Attacks. In Figure 5, we compare the backdoor task and main task accuracy during FL training under our RL-based attack, DBA, and PGD attack, when Krum or Median is considered as the defense mechanism. Among the three attacks, the RL attack reaches $\sim 100\%$ backdoor accuracy within 100 FL rounds. The backdoor accuracy of DBA is unstable due to the introduction of subsampling, which is ignored in the original paper. Further, its attack performance stays at a low level during the whole FL training in our setting. This is mainly because each DBA attacker uses a randomly sampled sub-trigger (see Figure 3) while baseline attackers all use the same global trigger. The latter is more powerful given the same number of attackers. Among the three baselines, PGD achieves relatively better performance compared with BFL (see Figure 1) and DBA. However, our RL-based attack still outperforms the PGD attack as it utilizes adaptive actions based on a long-term objective. The attack performance of all attacks typically grows faster under Krum compared with Median due to the fact that under Krum, an attacker's input will completely replace the actual global model once it is chosen by the server. Figure 6(a) gives similar results under the norm-bounding defense. Compared with DBA, PGD attack achieves better backdoor accuracy but results in relatively lower main task accuracy, due to the fixed projection bound and other hyperparameters chosen. Our RL attack outperforms all baselines for both backdoor task and main task accuracy.

Additional Durability Result. Figure 6(b) shows the accuracy of both backdoor and main tasks when the attacker only attacks during the first 100 FL rounds. We observe that Anticipate achieves better backdoor durability than BFL and Neorontoxin but at the cost of a slower growth rate of the main task accuracy. Our RL attack obtains the highest backdoor durability while maintaining good main task accuracy. The latter can be further improved by tuning λ and the discount factor γ during training.

C.2 Ablation Studies

Impact of λ. Figure 7(a) shows the backdoor and main task accuracy when $\lambda = 0$. Compared with Figure 2(a) where $\lambda = 0.5$, we observe that the backdoor accuracy for the RL attack further improves as expected while the main task accuracy becomes significantly worse as it is completely ignored in the reward function. On the other hand, the performance of BFL is independent of λ (as it only uses the poison ratio to obtain such a tradeoff). This indicates the importance of choosing

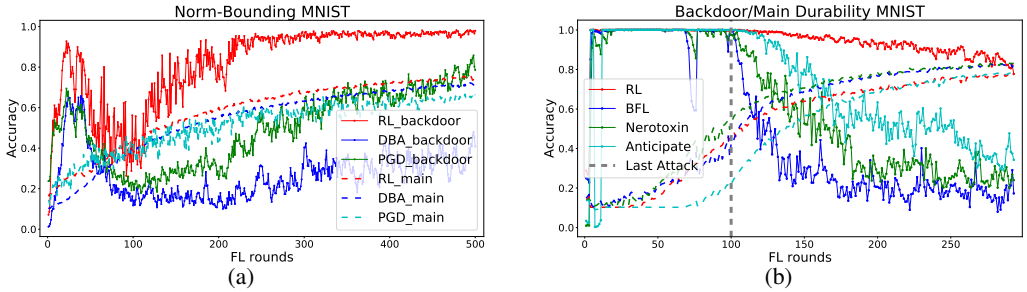

Figure 6: (a) A comparison of the global model accuracy for the backdoor task and the main task during FL training, under RL attack, BFL, PGD attack, and norm-clipping defense. (b) A comparison of backdoor attack durability (solid line) and the corresponding main task accuracy (dashed line) of the RL attack, BFL, Neurotoxin, and Anticipate. Attacks are applied during the first 100 FL rounds only.

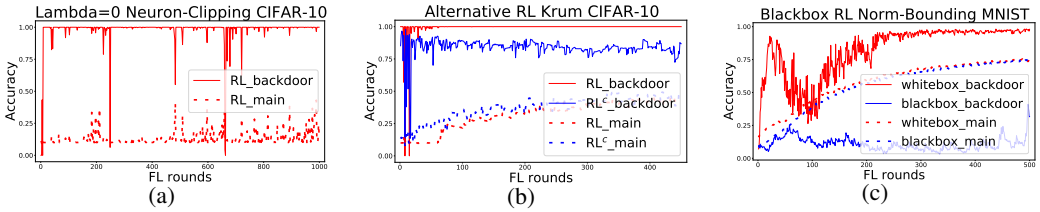

Figure 7: A comparison of the global model accuracy for the backdoor task and the main task during FL training, under (a) RL attack, BFL, and post-training Neuron Clipping defense; (b) RL attacks based on alternative training (RL) and simultaneous training (RL^c), where the lengths of training are both 80,000 steps, and Krum defense; (c) white-box vs. black-box RL attack under the norm-bounding defense.

a proper λ (or poison ratio) to maximize the backdoor attack performance while maintaining good main task accuracy. As it is time consuming to search for such a λ manually, a promising direction is to consider constrained RL with the desired main task accuracy as a constraint, which is left to our future work.

Efficiency of Alternative Training. Figure 7(b) compares a policy trained using the alternative training approach (called RL) and a policy trained using the simultaneous training approach (called RL^c), where both policies are trained for 80,000 time steps in total. For alternative training, we train the local search policy for 10,000 steps and the model crafting policy for 10,000 steps in each iteration, with 80,000 total training steps (i.e., the number of training iterations is 4). We observe that the former achieves nearly 100% backdoor accuracy while the latter only reaches a backdoor accuracy around 80%, indicating that the simultaneous training does not converge over $80,000$ steps.

Importance of Knowing FL System Parameters. In this work, we focus on the commonly considered white-box attack setting and assume that the RL attacker has prior knowledge about the FL environment, including the server's defense mechanism, the local training method, the number of devices, and the subsampling rate. Figure 7(c) gives the result of a naive black-box RL attack, where the system adopts the norm-bounding defense (with the default clipping bound), while the attacker does not know that and instead assumes that FedAvg is used as the aggregation rule and there is no post-training defense. As shown in the figure, the black-box attack fails to compromise the FL system. This is because with the wrong "world model", the attacker tends to learn an aggressive attack policy, resulting in malicious model updates that are significantly different from benign updates and can be easily filtered out by the defense. Intuitively, the attacker should be more conservative in the black-box setting by considering a worst-case defense scenario so that it could achieve certain level of attack performance even under weaker defenses. Further, the attacker should constantly adapt to the unknown environment by updating its policy using the real-time feedback obtained during online FL training. How to design an effective RL-based backdoor attack in the black-box setting is an interesting open problem and is left to our future work.