Poisoning Attacks and Defenses to Federated Unlearning

Wenbin Wang* ShanghaiTech University Shanghai, China

Yuchen Liu North Carolina State University Raleigh, USA Qiwen Ma* Xidian University Xi'an, China

Zhuqing Liu University of North Texas Denton, USA Zifan Zhang North Carolina State University Raleigh, USA

> Minghong Fang University of Louisville Louisville, USA

Abstract

Federated learning allows multiple clients to collaboratively train a global model with the assistance of a server. However, its distributed nature makes it susceptible to poisoning attacks, where malicious clients can compromise the global model by sending harmful local model updates to the server. To unlearn an accurate global model from a poisoned one after identifying malicious clients, federated unlearning has been introduced. Yet, current research on federated unlearning has primarily concentrated on its effectiveness and efficiency, overlooking the security challenges it presents. In this work, we bridge the gap via proposing BadUnlearn, the first poisoning attacks targeting federated unlearning. In BadUnlearn, malicious clients send specifically designed local model updates to the server during the unlearning process, aiming to ensure that the resulting unlearned model remains poisoned. To mitigate these threats, we propose UnlearnGuard, a robust federated unlearning framework that is provably robust against both existing poisoning attacks and our BadUnlearn. The core concept of UnlearnGuard is for the server to estimate the clients' local model updates during the unlearning process and employ a filtering strategy to verify the accuracy of these estimations. Theoretically, we prove that the model unlearned through UnlearnGuard closely resembles one obtained by train-from-scratch. Empirically, we show that BadUnlearn can effectively corrupt existing federated unlearning methods, while UnlearnGuard remains secure against poisoning attacks.

CCS Concepts

• Security and privacy → Systems security.

Keywords

Federated Unlearning, Poisoning Attacks, Robustness

ACM Reference Format:

Wenbin Wang, Qiwen Ma, Zifan Zhang, Yuchen Liu, Zhuqing Liu, and Minghong Fang. 2025. Poisoning Attacks and Defenses to Federated Unlearning. In Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3701716.3715494

*Equal contribution. Wenbin Wang and Qiwen Ma conducted this research while they were interns under the supervision of Minghong Fang.



This work is licensed under a Creative Commons Attribution International 4.0 License.

WWW Companion '25, April 28-May 2, 2025, Sydney, NSW, Australia © 2025 Copyright held by the owner/author(s). ACM ISBN 979-8-4007-1331-6/25/04 https://doi.org/10.1145/3701716.3715494

1 Introduction

Federated learning (FL) [12] allows clients to collaboratively train models while respecting privacy. However, its decentralized nature introduces vulnerabilities to poisoning attacks [6, 14, 19, 20]. While Byzantine-robust aggregation methods [3, 7-10, 18] mitigate such attacks, they remain susceptible to sophisticated strategies. As a result, researchers have proposed various detection-based methods [4, 13, 17] to identify malicious clients in the FL system. Despite their potential, these methods cannot fully prevent poisoning, as the global model may already be compromised. To eliminate the impact of malicious clients on the final learned global model, one obvious approach is to retrain the model entirely from scratch after excluding malicious clients, a process often referred to as trainfrom-scratch; however, this approach is computationally inefficient. Recognizing these inefficiencies, federated unlearning (FU) mechanisms [5, 11, 16] have been designed to efficiently remove the influence of corrupted models. While numerous studies have focused on enhancing the utility of the FU process, such as improving its effectiveness and efficiency, the security concerns associated with FU have been largely overlooked.

Our work: In this work, we bridge this gap by introducing BadUnlearn, the first poisoning attacks specifically designed to target FU. Note that in current FU methods, the server requests clients to calculate exact local model updates and transmit these updates back during specific training rounds to achieve a more accurate global model. This communication between the server and clients during the FU process creates an opportunity for attackers to exploit. In our proposed approach, BadUnlearn, malicious clients send meticulously crafted harmful local model updates to the server throughout the FU process. This manipulation aims to ensure that the final unlearned model remains similar to the poisoned model (referred to as the learned model) obtained during the FL phase. In other words, BadUnlearn seeks to cause existing FU methods to fail by preventing them from successfully unlearning an accurate global model from the poisoned one. We frame our attack as an optimization problem, where malicious updates depend on the server's aggregation rules. However, solving this is challenging due to non-differentiable aggregation methods like Median [18] in FU techniques such as FedRecover [5]. To address this, we model each malicious update as a perturbed version of the learned model.

To counter the BadUnlearn attack, we propose a secure FU framework, UnlearnGuard. The core idea of UnlearnGuard is that the server stores historical data, including clients' local model updates and global models, from the FL process. This data helps estimate local model updates during the FU phase. However, these estimations

may not always be accurate, so we introduce a filtering strategy to verify them. The strategy ensures that an estimated update does not significantly deviate from the corresponding stored update. The first variant of our robust FU method, UnlearnGuard-Dist, considers an estimated update accurate if it is close in distance to the stored update. However, UnlearnGuard-Dist overlooks the direction of updates, allowing the attacker to manipulate the method with small magnitude updates. To address this, we introduce UnlearnGuard-Dir, where an estimated update is considered precise if it matches the direction of the original update from the FL phase. We summarize our main contributions in this paper as follows:

- We present BadUnlearn attack, the first poisoning attacks targeting the FU process.
- We introduce UnlearnGuard, a FU framework that withstands both existing poisoning attacks and our BadUnlearn. We theoretically demonstrate that, under mild assumptions, the model unlearned by UnlearnGuard closely approximates the one trained from scratch.
- Extensive experiments across various poisoning attacks, aggregation rules, and FU methods demonstrate that BadUnlearn effectively compromises existing FU methods, whereas our proposed UnlearnGuard remains robust.

2 Preliminaries and Threat Model

2.1 Federated Learning (FL)

In FL, n clients collaboratively train a global model \mathbf{w} without sharing local dataset D_i , where D_i is client i's training data, $i \in [n]$, and [n] represent the set $\{1,2,\ldots,n\}$. The model minimizes the loss $\mathcal{L}(\mathbf{w},D) = \sum_{i \in [n]} \frac{|D_i|}{|D|} \mathcal{L}_i(\mathbf{w},D_i)$, where $D = \bigcup_{i \in [n]} D_i$. At the training round t, the server broadcasts \mathbf{w}^t to all clients (Step I), clients train locally and compute updates \mathbf{g}_i^t (Step II), and the server aggregates updates using ARR to update $\mathbf{w}^{t+1} = \mathbf{w}^t - \eta$. ARR $\left\{\mathbf{g}_i^t: i \in [n]\right\}$ (Step III), where ARR is the aggregation rule, η is the learning rate.

2.2 Federated Unlearning (FU)

FU [5, 15, 16] aims to mitigate the influence of malicious clients after a global model is trained, focusing on removing entire clients rather than individual data points, as in centralized learning. Existing research on FU has primarily focused on improving utility and accuracy, neglecting security challenges. FU seeks to eliminate the effects of compromised clients, which can introduce manipulated updates, but current techniques largely overlook the risks of poisoning attacks. To address these concerns, more research is needed to ensure FU methods are robust against malicious clients.

2.3 Threat Model

Attacker's goal and knowledge: We consider a scenario where an attacker controls malicious clients. During the FU process, these clients send crafted model updates to ensure the unlearned model remains similar to the model learned from the FL process. We consider three settings: full-knowledge, where the attacker knows all clients' updates and the aggregation rule ARR; partial-knowledge, where the attacker knows only the updates of malicious clients and

the ARR; and *black-box*, where the attacker knows only the updates of malicious clients and is unaware of the server's aggregation rule.

Defender's goal: Our goal is to develop an effective unlearning method with two key objectives: (i) robustness, ensuring resilience to poisoning attacks during the FU process, and (ii) aggregation rule independence, allowing compatibility with any server aggregation rule. Moreover, the method assumes the server remains unaware of both the attacks and the number of malicious clients.

3 Our BadUnlearn

In existing FU methods like FedRecover, the server collects and aggregates clients' local model updates during specific rounds. However, this communication allows malicious clients to introduce harmful updates, which can undermine the FU process. In our proposed BadUnlearn attack, the attacker strategically craft updates to ensure the resulting unlearned model remains close to the final learned model from the FL phase. This attack can be formulated as an optimization problem:

$$\min \| {}^{\#}\mathbf{w} - \mathsf{ARR}\{\mathbf{g}_{i}^{t} : i \in [n]\} \|, \tag{1}$$

where ${}^{\#}\mathbf{w}$ is the final learned model from the FL phase, ARR $\{\mathbf{g}_i^t: i \in [n]\}$ denotes the aggregated model update after the attack during the FU process, $\|\cdot\|$ denotes the ℓ_2 norm.

Solving this is challenging due to the non-differentiability of most aggregation rules. To overcome this, we approximate the solution by assuming each malicious update is a perturbed version of the pre-attack aggregated model ${}^{\#}\mathbf{w}$. Specifically, $\mathbf{g}_k^t = {}^{\#}\mathbf{w} + \epsilon \psi, k \in \mathcal{B}$, where ϵ is a scaling factor, and ψ is a perturbation vector, \mathcal{B} represent the set of malicious clients. We can define ψ as a predefined parameter, for example, $\psi = -\text{sign}({}^{\#}\mathbf{w})$, where sign represents the coordinate-wise sign of ${}^{\#}\mathbf{w}$, thereby simplifying the optimization problem to:

$$\underset{\epsilon}{\arg\max} - \| ^{\#}\mathbf{w} - \mathsf{ARR}\{\mathbf{g}_{i}^{t} : i \in [n]\} \|$$
s.t.
$$\mathbf{g}_{k}^{t} = ^{\#}\mathbf{w} + \epsilon \psi, \quad \psi = -\mathrm{sign}(^{\#}\mathbf{w}), \quad k \in \mathcal{B}.$$
(2)

The attacker determines ϵ heuristically, adjusting it based on changes in the objective function. Once ϵ is set, malicious updates are sent to the server, disrupting the FU process.

4 UnlearnGuard

4.1 Overview

In our proposed UnlearnGuard, the server retains historical data from the FL process. After excluding malicious clients, it uses FU to unlearn and rebuild a clean global model. Starting with a new initialization, the server iteratively trains the model as outlined in Section 2.1, estimating client updates from stored data and requesting the clients to compute the exact updates when necessary.

4.2 Model Updates Prediction

In our UnlearnGuard, the server uses historical data from the FL process to estimate client updates during FU. Let $\tilde{\mathbf{g}}_i^t$ and $\tilde{\mathbf{w}}^t$ denote the local and global model updates at round t during FL, stored by the server. In FU, \mathbf{g}_i^t denotes the exact update (calculated using local training data), while $\bar{\mathbf{g}}_i^t$ represents the estimated update for

client i. Using the Cauchy mean value theorem, $\bar{\mathbf{g}}_{i}^{t}$ is given by:

$$\bar{\mathbf{g}}_{i}^{t} = \tilde{\mathbf{g}}_{i}^{t} + \mathbf{H}_{i}^{t}(\mathbf{w}^{t} - \tilde{\mathbf{w}}^{t}), \tag{3}$$

where \mathbf{H}_i^t , the Hessian matrix for client i, is calculated as $\mathbf{H}_i^t = \int_0^1 \mathbf{H}(\tilde{\mathbf{w}}^t + y(\mathbf{w}^t - \tilde{\mathbf{w}}^t)) dy$. Direct computation of \mathbf{H}_i^t is computationally expensive; hence, the L-BFGS algorithm approximates it using limited historical data. Define the global model differences as $\Delta \mathbf{w}^t = \mathbf{w}^t - \tilde{\mathbf{w}}^t$, and the local update differences for client i at training round t as $\Delta \mathbf{g}_i^t = \tilde{\mathbf{g}}_i^t - \tilde{\mathbf{g}}_i^t$. For the last s rounds, let $\Delta \mathbf{W}^{t,s} = \{\Delta \mathbf{w}^{t-s}, \ldots, \Delta \mathbf{w}^{t-1}\}$ and $\Delta \mathbf{G}_i^{t,s} = \{\Delta \mathbf{g}_i^{t-s}, \ldots, \Delta \mathbf{g}_i^{t-1}\}$. L-BFGS uses $\Delta \mathbf{W}^{t,s}$, $\Delta \mathbf{G}_i^{t,s}$, and $\Delta \mathbf{w}^t$ to compute Hessian-vector products $\mathbf{H}_i^t(\mathbf{w}^t - \tilde{\mathbf{w}}^t)$, which are then used to predict $\tilde{\mathbf{g}}_i^t$.

4.3 UnlearnGuard-Dist

In Section 4.2, we demonstrate that the server can estimate client updates during FU using historical data, though inaccuracies may arise and degrade model quality over time. To address this, we propose UnlearnGuard-Dist, a distance-based calibration technique to minimize estimation errors. The central idea of UnlearnGuard-Dist is that, during a specific training round, if a client's local model update estimated in the FU process significantly deviates from the corresponding local model update stored during the FL process, it indicates an inaccurate estimation. Specifically, we consider the estimated update $\tilde{\mathbf{g}}_t^t$ accurate if it satisfies the following condition:

$$\max_{t_1 \in [t-r,t]} \left\| \bar{\mathbf{g}}_i^t - \tilde{\mathbf{g}}_i^{t_1} \right\| \le \max_{t_2,t_3 \in [t-r,t]} \left\| \tilde{\mathbf{g}}_i^{t_2} - \tilde{\mathbf{g}}_i^{t_3} \right\|, \tag{4}$$

where r is the buffer parameter. The left term measures the largest deviation of $\bar{\mathbf{g}}_i^t$ from the stored updates, while the right term represents the largest deviation among the stored updates themselves. This ensures that $\bar{\mathbf{g}}_i^t$ aligns with the client's historical update behavior. If this condition is met, the server uses $\bar{\mathbf{g}}_i^t$ to compute the global model for round t. Otherwise, it requests the exact update \mathbf{g}_i^t from the client to ensure reliability.

4.4 UnlearnGuard-Dir

Our proposed UnlearnGuard-Dist uses distance-based calibration to ensure the accuracy of estimated client updates, focusing on their magnitude. However, UnlearnGuard-Dist overlooks update directions, which can result in suboptimal unlearning even when the distance condition in Eq. (4) is satisfied. To address this, we introduce UnlearnGuard-Dir, a direction-aware calibration method that considers both magnitude and direction to reduce estimation errors during FU. In UnlearnGuard-Dir, an estimated update $\bar{\mathbf{g}}_i^t$ is deemed accurate if it satisfies:

$$\max_{t_1 \in [t-r,t]} \cos(\tilde{\mathbf{g}}_i^t, \tilde{\mathbf{g}}_i^{t_1}) \le \max_{t_2, t_3 \in [t-r,t]} \cos(\tilde{\mathbf{g}}_i^{t_2}, \tilde{\mathbf{g}}_i^{t_3}), \tag{5}$$

where $\cos(\cdot, \cdot)$ measures cosine similarity. This ensures $\bar{\mathbf{g}}_i^t$ aligns with the trend of historical updates without deviation. If the condition holds, we rescale $\bar{\mathbf{g}}_i^t$ to match the typical magnitude of past updates using $\bar{\mathbf{g}}_i^t \leftarrow \frac{\bar{\mathbf{g}}_i^t}{\|\bar{\mathbf{g}}_i^t\|} \times \operatorname{Median}\{\|\tilde{\mathbf{g}}_i^{t-r}\|, \dots, \|\tilde{\mathbf{g}}_i^t\|\}$, where the median is computed over the magnitudes of updates from rounds t-r to t. Algorithm 1 provides the pseudocode of UnlearnGuard, assuming n clients, with m malicious clients removed after detection, leaving n-m for FU.

Algorithm 1 The UnlearnGuard method.

```
Input: Set of clients in the FU phase S = \{i \mid m+1 \le i \le n\}; global models and model updates information in FL phase \tilde{\mathbf{w}}^0, \tilde{\mathbf{w}}^1, \dots, \tilde{\mathbf{w}}^T; \tilde{\mathbf{g}}_i^0, \tilde{\mathbf{g}}_i^1, \dots, \tilde{\mathbf{g}}_i^{T-1}; learning rate \eta; L-BFGS buffer size s; aggregation rule ARR. Output: Unlearned model \mathbf{w}^T.
```

```
1: Initialize \mathbf{w}_0 \leftarrow \tilde{\mathbf{w}}^0.
  2: for t = 0 to T - 1 do
                      if t < r then
  4:
                                 The server broadcasts \mathbf{w}_t to the clients in S.
                                 // Each client trains its local model and computes model
  5:
          update
                                 Each client i computes \mathbf{g}_i^t = \nabla \mathcal{L}_i(\mathbf{w}^t).
  6:
                                 // Aggregation and global model updating.
  7:
                                 \mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \cdot \mathsf{ARR}(\mathbf{g}_{m+1}^t, \dots, \mathbf{g}_n^t).
  8:
  9:
                                 Update L-BFGS buffers \Delta \mathbf{W}^{t,s}, \Delta \mathbf{G}^{t,s}.
10:
                                 \mathbf{for}\ i = m + 1 \text{ to } n \mathbf{\ do}
11:
                                            \Delta \mathbf{w}^t = \mathbf{w}^t - \tilde{\mathbf{w}}^t
12:
13:
                                            // We denote diag(X) as the diagonal matrix of X,
         and tril(X) the lower triangular matrix of X. \mathbf{A}_i^{t,s} = (\Delta \mathbf{W}^{t,s})^\top \Delta \mathbf{G}_i^{t,s}, \mathbf{D}_i^{t,s} = \mathrm{diag}(\mathbf{A}_i^{t,s})
14:
                                         \begin{aligned} \mathbf{A}_{i}^{t} &= (\Delta \mathbf{w}^{t})^{\top} \Delta \mathbf{G}_{i}^{t}, \mathbf{D}_{i}^{t} \quad \Delta \mathbf{G}_{i-1}^{t} \\ \mathbf{L}_{i}^{t,s} &= \operatorname{tril}(\mathbf{A}_{i}^{t,s}) \\ \sigma &= ((\Delta \mathbf{g}_{i}^{t-2})^{\top} \Delta \mathbf{w}^{t-2}) / ((\Delta \mathbf{w}^{t-2})^{\top} \Delta \mathbf{w}^{t-2}) \\ \mathbf{p} &= \begin{bmatrix} -\mathbf{D}_{i}^{t,s} & (\mathbf{L}_{i}^{t,s})^{\top} & (\Delta \mathbf{G}_{i}^{t,s})^{\top} \Delta \mathbf{w}^{t} \\ \mathbf{L}_{i}^{t,s} & \sigma(\Delta \mathbf{W}^{t,s})^{\top} \Delta \mathbf{W}^{t,s} \end{bmatrix}^{-1} \begin{bmatrix} (\Delta \mathbf{G}_{i}^{t,s})^{\top} \Delta \mathbf{w}^{t} \\ \sigma(\Delta \mathbf{W}^{t,s})^{\top} \Delta \mathbf{w}^{t} \end{bmatrix} \\ \mathbf{H}_{i}^{t} \Delta \mathbf{w}^{t} &= \sigma \Delta \mathbf{w}^{t} - \left[ \Delta \mathbf{G}_{i}^{t,s} & \sigma(\Delta \mathbf{W}^{t,s}) \right] \mathbf{p} \end{aligned}
15:
16:
17:
18
                                            \tilde{\mathbf{g}}_{i}^{t} = \tilde{\mathbf{g}}_{i}^{t} + \mathbf{H}_{i}^{t} \Delta \mathbf{w}^{t}
19:
                                            // Exact training
20:
                                            {f if} condition in Eq. (4) or Eq. (5) is not met {f then}
21:
                                                       Compute \mathbf{g}_i^t = \nabla \mathcal{L}_i(\mathbf{w}^t), and update \bar{\mathbf{g}}_i^t \leftarrow \mathbf{g}_i^t
          (rescale \bar{\mathbf{g}}_{i}^{t} if necessary).
23:
24
                                 \mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \cdot \mathsf{ARR}(\bar{\mathbf{g}}_{m+1}^t, \dots, \bar{\mathbf{g}}_n^t)
25
                      end if
26
27: end for
```

4.5 Security Analysis

Assumption 1. The loss function \mathcal{L}_i for client i is μ -strongly convexity and L-smoothness. Specifically, the following inequalities hold for any vectors $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d$:

$$\mathcal{L}_{i}(\mathbf{w}_{1}) \geq \mathcal{L}_{i}(\mathbf{w}_{2}) + \nabla \mathcal{L}_{i}(\mathbf{w}_{2})^{\top}(\mathbf{w}_{1} - \mathbf{w}_{2}) + \frac{\mu}{2} \|\mathbf{w}_{1} - \mathbf{w}_{2}\|^{2},$$

$$\mathcal{L}_{i}(\mathbf{w}_{1}) \leq \mathcal{L}_{i}(\mathbf{w}_{2}) + \nabla \mathcal{L}_{i}(\mathbf{w}_{2})^{\top}(\mathbf{w}_{1} - \mathbf{w}_{2}) + \frac{L}{2} \|\mathbf{w}_{1} - \mathbf{w}_{2}\|^{2}.$$

Assumption 2. The L-BFGS algorithm bounds the error in approximating Hessian-vector products as $\|\mathbf{H}_i^t(\mathbf{w}^t - \tilde{\mathbf{w}}^t) + \tilde{\mathbf{g}}_i^t - \bar{\mathbf{g}}_i^t\| \le M$, for any i and t, where M is a finite positive constant.

Theorem 1. Under Assumptions 1 and 2, with a learning rate $\eta \leq \min(\frac{1}{\mu}, \frac{1}{L})$ and effective detection of all malicious clients during FL, the discrepancy between the global model unlearned via UnlearnGuard-Dist (or UnlearnGuard-Dir) and the train-from-scratch

Table 1: Results of various aggregation rules under attacks where the attacker manipulates the FL process on MNIST.

ſ	Dataset	Attack FL	FedAvg	Median	TrMean
ĺ		No attack	0.04	0.06	0.06
	MNIST	Trim attack	0.23	0.41	0.20
- 1		Backdoor attack	0.02 / 0.99	0.09 / 0.01	0.06 / 0.01

Table 2: Results of Train-from-scratch and Historical methods on MNIST dataset, where the attacker manipulates the FL process, and the server aims to obtain a clean global model.

Attack FL	ARR	Train-from-scratch	Historical
	FedAvg	0.05	0.90
Trim attack	Median	0.07	0.90
	TrMean	0.06	0.90
	FedAvg	0.05 / 0.00	0.85 / 0.00
Backdoor attack	Median	0.06 / 0.00	0.89 / 0.00
	TrMean	0.06 / 0.00	0.86 / 0.00

model in each round t > 0 is bounded as:

$$\|\mathbf{w}^t - \ddot{\mathbf{w}}^t\| \le (\sqrt{1 - \eta \mu})^t \|\mathbf{w}^0 - \ddot{\mathbf{w}}^0\| + \frac{1 - (\sqrt{1 - \eta \mu})^t}{1 - \sqrt{1 - \eta \mu}} \eta M$$

where \mathbf{w}^t and $\ddot{\mathbf{w}}^t$ are the global models from UnlearnGuard-Dist (or UnlearnGuard-Dir) and train-from-scratch, respectively, in round t.

Proof. See Appendix A for the proof.

5 Experiments

5.1 Experimental Setup

Due to space constraints, we only use the MNIST dataset. For aggregation during FL training and unlearning, we mainly use FedAvg [12], Median [18], and Trimmed-mean (TrMean) [18], while also evaluating Bulyan [10], Krum [3], and FLAME [13]. We examine three attack schemes: Trim attack [6], which disrupts FL by crafting malicious updates to distort global models; Backdoor attack [1], where triggers embedded in training data lead to targeted malicious outcomes; and our BadUnlearn, aimed to manipulate FU process. Full-knowledge attack setting is considered by default.

- 5.1.1 FL and FU Settings. Training involved 2,000 rounds for MNIST (learning rate 3×10^{-4} , batch size 32). A CNN architecture consisting of two convolutional layers, each paired with a pooling layer, followed by two fully connected layers. We assume 100 clients, with 20% being malicious. FU follows FL settings and uses 80 clients, with 20% performing attacks like Trim attack or BadUnlearn attack. Malicious clients detected in FL are removed during FU. The buffer parameter is r=5, assuming Trim attack in FL and BadUnlearn attack in FU. For MNIST, we simulate the Non-IID distribution using the approach in [6]. The Non-IID degree is set to 0.5 following [6].
- 5.1.2 Evaluation Metrics. We consider two metrics: testing error rate (TER) and attack success rate (ASR). Higher TER and ASR indicate stronger attacks, lower values reflect better defenses.
- 5.1.3 Compared Methods. We compare UnlearnGuard with three unlearning methods: Train-from-scratch, Historical-information-only (Historical) [5], and FedRecover [5]. Historical method rebuilds the global model using stored updates from the FL process, refining it incrementally without communication with client during FU.

5.2 Experimental Results

BadUnlearn compromises FU methods, while UnlearnGuard remains robust: Table 1 presents the performance of aggregation

Table 3: Results of FU methods on the MNIST dataset, where the attacker manipulates both FL and FU processes.

Attack FL	Attack FU	ARR	FedRecover	UnlearnGuard-Dist	UnlearnGuard-Dir
		FedAvg	0.06	0.05	0.05
	No attack	Median	0.05	0.06	0.04
	1	TrMean	0.07	0.07	0.04
		FedAvg	0.05	0.06	0.06
No attack	Trim attack	Median	0.12	0.12	0.08
		TrMean	0.15	0.11	0.06
		FedAvg	0.06	0.05	0.05
	BadUnlearn	Median	0.07	0.11	0.07
		TrMean	0.06	0.13	0.05
		FedAvg	0.05	0.06	0.05
	No attack	Median	0.11	0.14	0.09
		TrMean	0.15	0.12	0.06
	Trim attack	FedAvg	0.06	0.07	0.06
Trim attack		Median	0.15	0.14	0.10
		TrMean	0.16	0.15	0.07
	BadUnlearn	FedAvg	0.24	0.07	0.06
		Median	0.39	0.14	0.09
		TrMean	0.23	0.14	0.06
	No attack	FedAvg	0.05 / 0.00	0.02 / 0.00	0.02 / 0.00
		Median	0.12 / 0.01	0.11 / 0.01	0.10 / 0.00
		TrMean	0.07 / 0.00	0.06 / 0.00	0.06 / 0.00
Backdoor		FedAvg	0.06 / 0.01	0.03 / 0.01	0.03 / 0.00
attack	Trim attack	Median	0.12 / 0.00	0.12 / 0.00	0.11 / 0.00
attack		TrMean	0.04 / 0.00	0.06 / 0.00	0.05 / 0.00
		FedAvg	0.04 / 0.98	0.03 / 0.01	0.03 / 0.00
	BadUnlearn	Median	0.10 / 0.02	0.13 / 0.00	0.12 / 0.00
		TrMean	0.08 / 0.03	0.07 / 0.00	0.07 / 0.00

Table 4: Partial-knowledge and black-box attacks.

	Partial-knowledge			Black-box		
ARR	FedRecover	UnlearnGuard	UnlearnGuard	FedRecover	UnlearnGuard	UnlearnGuard
		-Dist	-Dir		-Dist	-Dir
FedAvg	0.23	0.07	0.06	0.22	0.06	0.05
Median	0.38	0.14	0.09	0.38	0.14	0.08
TrMean	0.23	0.14	0.06	0.23	0.13	0.05

Table 5: The server employs advanced rules for FL and FU.

	ARR	Learned model [#] w	FedRecover	UnlearnGuard-Dist	UnlearnGuard-Dir
	Bulyan	0.13	0.19	0.12	0.13
	Krum	0.10	0.16	0.11	0.12
ĺ	FLAME	0.06	0.21	0.09	0.08

Table 6: Malicious clients execute advanced attacks in FL.

Γ	Attack	Learned model #w	FedRecover	UnlearnGuard-Dist	UnlearnGuard-Dir
Γ	Min-Max	0.22	0.20	0.13	0.12
Г	Min-Sum	0.23	0.24	0.12	0.09
Г	LIE	0.18	0.19	0.09	0.08

strategies under attacks during FL on the MNIST dataset. "Attack FL" means malicious clients perform attack during FL. Backdoor attack results are presented as "TER / ASR", while only TER is reported for other attacks. "No attack" means all clients are benign. Table 2 illustrates the results of Train-from-scratch and Historical methods. We observe from Table 1 that aggregation strategies like FedAvg, Median, and TrMean are vulnerable to poisoning attacks, while Table 2 indicates the Historical method fails to recover a clean global model when FL is compromised.

Table 3 presents the results of the FedRecover, UnlearnGuard-Dist, and UnlearnGuard-Dir methods on the MNIST dataset. "Attack FU" means malicious clients perform attack during FU. Based on Tables 1-3, we can draw several observations. BadUnlearn effectively compromises FedRecover, a leading FU technique. Additionally, FedRecover demonstrates resilience against Trim attacks during the FU phase, consistent with findings in [5]. Furthermore, our UnlearnGuard-Dist and UnlearnGuard-Dir, exhibit robustness against various poisoning attacks and successfully unlearn a clean global model from a compromised one.

Partial-knowledge and black-box attacks: Table 4 presents the results of partial-knowledge and black-box attacks. In the black-box

Table 7: Results of Adaptive attack.

	ARR	UnlearnGuard-Dist	UnlearnGuard-Dir	
ſ	FedAvg	0.06	0.05	
ĺ	Median	0.16	0.08	
ĺ	TrMean	0.15	0.07	

setting, the attacker is unaware of the server's aggregation rule and uses the Median rule to generate malicious updates. The results demonstrate that even in practical scenarios like the black-box setting, BadUnlearn remains effective in compromising FedRecover.

Complex aggregation rules and advanced attacks: We examines scenarios with advanced aggregation methods like Bulyan [10], Krum [3], and FLAME [13], where the attacker applies Trim attack during FL and BadUnlearn attack during FU. The same aggregation method is consistently applied across both FL and FU. Note that FLAME is an advanced method that can detect malicious clients. As shown in Table 5, our unlearning methods effectively mitigate poisoning attacks, even under these sophisticated aggregation strategies. We also evaluate cases where clients execute advanced attacks during FL, such as Min-Max [14], Min-Sum [14], and LIE [2], while continuing to employ BadUnlearn during FU. Results in Table 6 show that FedRecover remains vulnerable to BadUnlearn.

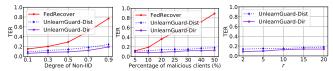
Adaptive attack: We evaluate UnlearnGuard against a challenging Adaptive attack [14], where the attacker has full knowledge of the FU process, including clients' updates, the server's aggregation rule, and filtering strategies (e.g., Eq. (4) or Eq. (5)). Malicious clients apply the Trim attack during FL and the Adaptive attack during FU. As shown in Table 7, both UnlearnGuard-Dist and UnlearnGuard-Dir demonstrate resilience to these strong attacks.

Attack more complex FU methods: We examine a scenario where the server uses the advanced FU method, such as FATS [16] or TFUA [15]. The attacker performs Trim attack during FL, and our BadUnlearn attack during FU. The testing error rates for FATS and TFUA are 0.25 and 0.22, respectively, demonstrating that BadUnlearn effectively disrupts FATS and TFUA during the FU process.

Server storage overhead: Our UnlearnGuard stores historical data with a space complexity of O(d), requiring O(dnT) storage for d parameters, n clients, and T training rounds. In our experiments, it used up to 200 GB, manageable for modern servers.

Ablation studies: Fig. 1a illustrates the impact of Non-IID degree, with larger x-axis values indicating more Non-IID data (default: 0.5). Both UnlearnGuard-Dist and UnlearnGuard-Dir show similar increasing trends, but UnlearnGuard-Dir maintains a consistently lower TER, while FedRecover exhibits the highest TER, worsening with increasing Non-IID. Fig. 1b shows that as malicious clients increase from 5% to 50%, FedRecover's TER rises from 0.12 to 0.89, while UnlearnGuard remain robust. Fig. 1c shows the impact of buffer size *r* on UnlearnGuard, with TER remaining generally stable.

Discussions: Our attack differs from traditional FL attacks by targeting the FU phase to prevent effective unlearning, ensuring the unlearned model mirrors the learned one regardless of testing error. Unlike FL poisoning attacks that corrupt training, we aim to compromise unlearning. Furthermore, our attack is feasible. Malicious clients remain dormant during FL phase—they do not perform attacks and act like benign clients—thereby evading detection during FL. These clients then execute their attack during FU process. Note that the server can detect malicious clients during FU process.



(a) Degree of Non-IID. (b) Malicious fraction.

(c) Number of r.

Figure 1: Impact of degree of Non-IID, percentage of malicious clients during FU, and the number of r on MNIST.

6 Conclusion

We proposed BadUnlearn, a novel poisoning attack targeting FU, which evades detection and disrupts unlearning by subtly altering model updates. To counter this, we introduced UnlearnGuard-Dist and UnlearnGuard-Dir, robust unlearning methods proven secure against poisoning attacks both theoretically and empirically.

References

- [1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In AISTATS.
- [2] Gilad Baruch, Moran Baruch, and Yoav Goldberg. 2019. A little is enough: Circumventing defenses for distributed learning. In NeurIPS.
- [3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In NeurIPS.
- [4] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2021. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In NDSS.
- [5] Xiaoyu Cao, Jinyuan Jia, Zaixi Zhang, and Neil Zhenqiang Gong. 2023. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In IEEE Symposium on Security and Privacy.
- [6] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to Byzantine-robust federated learning. In USENIX Security Symposium.
- [7] Minghong Fang, Jia Liu, Neil Zhenqiang Gong, and Elizabeth S Bentley. 2022. Aflguard: Byzantine-robust asynchronous federated learning. In ACSAC.
- [8] Minghong Fang, Seyedsina Nabavirazavi, Zhuqing Liu, Wei Sun, Sun-dararaja Sitharama Iyengar, and Haibo Yang. 2025. Do We Really Need to Design New Byzantine-robust Aggregation Rules?. In NDSS.
- [9] Minghong Fang, Zifan Zhang, Prashant Khanduri, Jia Liu, Songtao Lu, Yuchen Liu, Neil Gong, et al. 2024. Byzantine-robust decentralized federated learning. In CCS
- [10] Rachid Guerraoui, Sébastien Rouault, et al. 2018. The hidden vulnerability of distributed learning in byzantium. In ICML.
- [11] Ziyao Liu, Yu Jiang, Jiyuan Shen, Minyi Peng, Kwok-Yan Lam, Xingliang Yuan, and Xiaoning Liu. 2024. A survey on federated unlearning: Challenges, methods, and future directions. In ACM Computing Surveys.
- [12] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In AISTATS.
- [13] Thien Duc Nguyen, Phillip Rieger, Roberta De Viti, Huili Chen, Björn B Brandenburg, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, et al. 2022. FLAME: Taming backdoors in federated learning. In USENIX Security Symposium.
- [14] Virat Shejwalkar and Amir Houmansadr. 2021. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In NDSS.
- [15] Xinyi Sheng, Wei Bao, and Liming Ge. 2024. Robust Federated Unlearning. In CIKM.
- [16] Youming Tao, Cheng-Long Wang, Miao Pan, et al. 2024. Communication efficient and provable federated unlearning. In VLDB.
- [17] Yueqi Xie, Minghong Fang, and Neil Zhenqiang Gong. 2024. FedREDefense: Defending against Model Poisoning Attacks for Federated Learning using Model Update Reconstruction Error. In ICML.
- [18] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In ICML.
- [19] Ming Yin, Yichang Xu, Minghong Fang, and Neil Zhenqiang Gong. 2024. Poisoning federated recommender systems with fake users. In The Web Conference.
- [20] Zifan Zhang, Minghong Fang, Jiayuan Huang, and Yuchen Liu. 2024. Poisoning Attacks on Federated Learning-based Wireless Traffic Prediction. In IFIP/IEEE Networking Conference.

A Proof of Theorem 1

Here, we establish the bound of $\|\mathbf{w}^t - \ddot{\mathbf{w}}^t\|$ between the global model obtained by UnlearnGuard-Dist (or UnlearnGuard-Dir) and the train-from-scratch method. First, we delineate the update rules for unlearning processes.

Estimation phase: we employ both gradient (or local model update) and Hessian information such that:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \sum_{i=m+1}^n \alpha_i [\mathbf{H}_i^t (\mathbf{w}^t - \tilde{\mathbf{w}}^t) + \tilde{\mathbf{g}}_i^t], \tag{6}$$

where $\alpha_i = \frac{|D_i|}{|D'|}$, $D' = \bigcup_{i=m+1}^n D_i$ denotes the joint dataset of the remaining n-m clients. For simplicity, we will adopt these notations in the subsequent analysis.

Exact training phase: we only use the exact gradient of each client such that:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \sum_{i=m+1}^n \alpha_i \mathbf{g}_i^t. \tag{7}$$

Additionally, we denote \mathbf{h}_i^t as the model update contributed by the client i model in the t round of the train-from-scratch strategy. The global model, updated in round t is given by:

$$\ddot{\mathbf{w}}^t = \ddot{\mathbf{w}}^{t-1} - \eta \sum_{i=m+1}^n \alpha_i \mathbf{h}_i^t. \tag{8}$$

Consider **Estimation** phase, where \mathbf{w}^t is updated according to Eq. (6). Based on Eqs. (7) and (8), we have:

$$\|\mathbf{w}^{t+1} - \ddot{\mathbf{w}}^{t+1}\| = \|(\mathbf{w}^t - \eta \sum_{i=m+1}^n \alpha_i \mathbf{g}_i^t) - (\ddot{\mathbf{w}}^t - \eta \sum_{i=m+1}^n \alpha_i \mathbf{h}_i^t)\|$$

$$= \|\mathbf{w}^t - \ddot{\mathbf{w}}^t - \eta \sum_{i=m+1}^n \alpha_i (\mathbf{g}_i^t - \mathbf{h}_i^t)\|, \tag{9}$$

Applying triangle inequality, we bound for the difference between \mathbf{w}^{t+1} and $\ddot{\mathbf{w}}^{t+1}$ as:

$$\|\mathbf{w}^{t+1} - \ddot{\mathbf{w}}^{t+1}\|$$

$$\leq \|\mathbf{w}^{t} - \ddot{\mathbf{w}}^{t} - \eta \sum_{i=m+1}^{n} (\alpha_{i} \|\mathbf{g}_{i}^{t} - \mathbf{h}_{i}^{t}\|)\|$$

$$+ \|\eta \sum_{i=m+1}^{n} \alpha_{i} \left[\mathbf{g}_{i}^{t} - \mathbf{H}_{i}^{t}(\mathbf{w}^{t} - \tilde{\mathbf{w}}^{t}) + \tilde{\mathbf{g}}_{i}^{t}\right]\|. \tag{10}$$

It remains to bound terms I and II, respectively.

$$\mathbf{I}^{2} = \|\mathbf{w}^{t} - \ddot{\mathbf{w}}^{t}\|^{2} - 2\eta \langle \mathbf{w}_{t} - \ddot{\mathbf{w}}_{t}, \sum_{i=m+1}^{n} \alpha_{i} (\mathbf{g}_{i}^{t} - \mathbf{h}_{i}^{t}) \rangle$$
$$+ \eta^{2} \|\sum_{i=m+1}^{n} \alpha_{i} (\mathbf{g}_{i}^{t} - \mathbf{h}_{i}^{t}) \|^{2}, \tag{11}$$

where $\langle \cdot, \cdot \rangle$ represents the inner product of two vectors.

By applying the Cauchy-Schwarz inequality, we derive:

$$\mathbf{I}^{2} \leq (\|\mathbf{w}^{t} - \ddot{\mathbf{w}}^{t}\|^{2} - \eta \sum_{i=m+1}^{n} \alpha_{i} \langle \mathbf{w}^{t} - \ddot{\mathbf{w}}^{t}, \mathbf{g}_{i}^{t} - \mathbf{h}_{i}^{t} \rangle)$$
$$- \eta \sum_{i=m+1}^{n} \alpha_{i} \langle \mathbf{w}^{t} - \ddot{\mathbf{w}}^{t}, \mathbf{g}_{i}^{t} - \mathbf{h}_{i}^{t} \rangle + \eta^{2} \sum_{i=m+1}^{n} \alpha_{i}^{2} \|\mathbf{g}_{i}^{t} - \mathbf{h}_{i}^{t}\|^{2}. \quad (12)$$

Given Assumption 1 and Assumption 2, we see the loss function \mathcal{L}_i is μ -strong convexity and L-smoothness. Utilizing inequalities for inner products, we obtain:

$$\mathbf{I}^{2} \leq (\|\mathbf{w}^{t} - \ddot{\mathbf{w}}^{t}\|^{2} - \eta \mu \|\mathbf{w}^{t} - \ddot{\mathbf{w}}^{t}\|^{2})$$
$$- \eta \sum_{i=m+1}^{n} (\frac{\alpha_{i}}{L} - \frac{\eta}{\alpha_{i}^{2}}) \|\mathbf{g}_{i}^{t} - \mathbf{h}_{i}^{t}\|^{2}. \tag{13}$$

If the learning rate η meets the condition $\eta \le \frac{1}{L} \le \frac{|D'|}{L \max_{i=m+1}^{n} |D_i|}$, we obtain:

$$\mathbf{I}^{2} \le (1 - \eta \mu) \|\mathbf{w}^{t} - \ddot{\mathbf{w}}^{t}\|^{2}. \tag{14}$$

Furthermore, we have:

$$\mathbf{I} \le \sqrt{1 - \eta \mu} \|\mathbf{w}^t - \ddot{\mathbf{w}}^t\|. \tag{15}$$

For term II, we can derive an upper bound based on Assumption 2 such that:

$$\mathbf{II} \le \eta M. \tag{16}$$

Substituting (15) and (16) into (10), we have:

$$\|\mathbf{w}^{t+1} - \ddot{\mathbf{w}}^{t+1}\| \le \mathbf{I} + \mathbf{II} \le \sqrt{1 - \eta \mu} \|\mathbf{w}^t - \ddot{\mathbf{w}}^t\| + \eta M.$$
 (17)

In the following, we consider **Exact training** case, which occurs when t < r and the conditions in Eq. (4) or Eq. (5) are not satisfied. Similar to the previous analysis, we obtain:

$$\|\mathbf{w}^{t+1} - \ddot{\mathbf{w}}^{t+1}\| = \|\mathbf{w}^t - \ddot{\mathbf{w}}^t - \eta \sum_{i=m+1}^n \alpha_i (\mathbf{g}_i^t - \mathbf{h}_i^t)\|.$$
 (18)

It is obvious that the RHS of above equation is what we defined earlier as I. Hence, we have:

$$\|\mathbf{w}^{t+1} - \ddot{\mathbf{w}}^{t+1}\| \le \sqrt{1 - \eta \mu} \|\mathbf{w}^t - \ddot{\mathbf{w}}^t\|.$$
 (19)

Combining two cases, we have:

$$\|\mathbf{w}^{t+1} - \ddot{\mathbf{w}}^{t+1}\| \le \sqrt{1 - \eta \mu} \|\mathbf{w}^t - \ddot{\mathbf{w}}^t\| + \eta M.$$
 (20)

By recursion, we have:

$$\|\mathbf{w}^t - \ddot{\mathbf{w}}^t\| \le (\sqrt{1 - \eta \mu})^t \|\mathbf{w}^0 - \ddot{\mathbf{w}}^0\| + \frac{1 - (\sqrt{1 - \eta \mu})^t}{1 - \sqrt{1 - \eta \mu}} \eta M,$$
 (21)

for any $t\geq 0$, where \mathbf{w}^0 and $\ddot{\mathbf{w}}^0$ are the initial parameter values in the FU phase and train-from-scratch, respectively. This formulation concludes that under appropriate conditions on η , the bound converges to $\frac{\eta M}{1-\sqrt{1-\eta\mu}}$ as $t\to\infty$, thereby confirming the stability of the learning algorithm.