BACKDOOR ECHOES: SAME MODEL, DIFFERENT SECURITY FUTURES IN FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning models can retain backdoors even after malicious clients leave and training continues with only benign participants. Existing studies primarily attribute such persistence to model parameters, overlooking server optimizer states that encode historical updates and influence future training. We show that server optimizer state can substantially alter post-attack backdoor behavior even when model parameters are identical. Through causal state interventions, we observe an average 8.82% attack-success-rate (ASR) gap across independently pretrained image classifiers. On Llama-2-7B, this gap reaches 85.99% after twenty benign rounds, while clean behavior remains nearly identical. Motivated by this state dependence, we develop DYNAPHANTOM, a history-aware attack that conditions malicious submissions on previous submissions to indirectly shape the hidden server state without accessing it. Our findings identify optimizer state as an overlooked source of persistent backdoor risk and show that restoring identical model weights does not necessarily restore the same security trajectory.

1 INTRODUCTION

Federated learning models may retain backdoors even after malicious clients leave and training continues with only benign participants. Existing work has largely studied this persistence through the model parameters themselves (Bagdasaryan et al., 2020; Zhang et al., 2022). These studies primarily characterize this residual risk through the model itself: given a compromised model checkpoint, they measure how its attack success rate evolves under subsequent benign training. This view implicitly treats the current model *parameters* as the primary state determining future backdoor behavior.

However, federated optimization can also retain server-side state that summarizes past updates and influences future training (Reddi et al., 2021). For example, FedAvgM (Federated Averaging with Server Momentum) accumulates aggregated client updates across rounds in a server-side momentum buffer, which is then used to update the global model (Hsu et al., 2019). In a related setting, state-intervention work shows that first-moment differences can leave parameters and outputs unchanged at the intervention point, while altering subsequent training dynamics (Xu, 2026). This raises a largely overlooked question: **how does server optimizer state affect a backdoor after attacks cease?**

To answer this question, we conduct a pilot study to isolate the backdoor-specific effect of optimizer state: whether changing the retained state alters subsequent backdoor behavior when the current model is held fixed. Specifically, we perform a causal audit of server optimizer state. Starting from the same image-classification model checkpoint, we construct paired branches that either preserve the momentum from an attacked run or replace it with momentum from a matched clean run. We hold all other conditions fixed and continue training with only benign clients. Across six independently pretrained models, the two branches exhibit an average **8.82-percentage-point** gap in attack success rate (ASR). This causal audit demonstrates that server optimizer state alone can substantially alter future backdoor behavior under controlled interventions.

However, a realistic attacker cannot directly access or modify the server momentum. It can only influence the server state indirectly through its submitted updates. This raises a practical question: *can an attacker exploit the identified state dependence using only its own submissions?* To investigate this question, we develop DYNAPHANTOM, a history-aware attack that conditions each malicious submission on both current local information and a summary of its previous submissions.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The motivation follows directly from the recursive nature of optimizer state (e.g., FedAvgM), where momentum evolves as a decayed accumulation of updates from previous rounds. Although an attacker cannot observe the complete server status, its own submissions constitute the known portion of the update history through which it has influenced this hidden state. DYNAPHANTOM therefore uses the submission history to coordinate the attacker-controlled contributions that are repeatedly injected into the momentum recursion.

We evaluate the optimizer-state effect on both image classification and language modeling, using ResNet-18 (He et al., 2016) on CIFAR-10 (Krizhevsky & Hinton, 2009) and Llama-2-7B (Touvron et al., 2023) on SST-2 (Socher et al., 2013). Across both settings, we observe substantially different backdoor trajectories under different optimizer states. For Llama-2-7B, after five rounds of benign training, the ASR gap between two branches with different optimizer states reaches approximately 40 percentage points despite nearly identical clean accuracy. After twenty benign rounds, this gap increases to **85.99%**, while their clean predictions remain nearly identical. These results show that optimizer-state-dependent backdoor behavior can persist and increasingly diverge during benign continuation, even when the models remain nearly indistinguishable on clean inputs. Our findings also have practical implications for checkpoint restoration. Restoring the same model weights does not necessarily restore the same subsequent security trajectory if optimizer state is retained. Our contributions are summarized as follows:

- **We identify server optimizer state as an overlooked source of backdoor risk.** Through causal state interventions, we show that different server momentum can lead to substantially different backdoor trajectories even when training starts from identical model weights. Across six independently pretrained image classifiers, changing only the optimizer state produces an average ASR gap of **8.82 percentage points** under subsequent benign training.
- **We develop DYNAPHANTOM, a history-aware attack that exploits hidden server state without directly accessing it.** DYNAPHANTOM conditions each malicious submission on the attacker's previous submissions, allowing the attacker to coordinate its contributions to the recursive evolution of server momentum. This demonstrates that optimizer-state dependence is not only observable under researcher-controlled interventions, but can also be exploited through realistic client-side submissions.
- **We demonstrate the persistence and practical implications of optimizer-state-dependent backdoors.** Across ResNet-18 and Llama-2-7B, we observe substantially different post-attack trajectories under different optimizer states while maintaining nearly identical clean behavior. On Llama-2-7B, the ASR gap reaches **85.99 percentage points** after twenty benign rounds. Our results further show that restoring identical model weights does not necessarily restore the same security trajectory, making optimizer-state handling an important consideration for checkpoint restoration and backdoor evaluation.

2 RELATED WORK

Federated backdoors and persistence. Malicious participants can implant targeted backdoors into collaboratively trained federated models (Bagdasaryan et al., 2020; Gu et al., 2019; Xie et al., 2025; Zhao et al., 2026). Recent studies further explore strategies to improve attack effectiveness and stealthiness. Chronic poisoning uses small incremental perturbations across rounds (Yu et al., 2024), while selection-aware scheduling coordinates malicious participation to enhance effectiveness and stealthiness (Bai et al., 2023). Subsequent work has studied how such backdoors persist after the attack phase under continued benign training. Neurotoxin improves persistence by targeting parameters that are less likely to be overwritten by benign updates (Zhang et al., 2022), while later studies further investigate mechanisms for maintaining malicious behavior over time (Dai & Li, 2023; Liu et al., 2024). Backdoor persistence has also been studied in continual learning (Guo et al., 2025) and graph foundation models (Luo et al., 2026). Cao et al. (2024) further links backdoor persistence to activation patterns. Together, these studies establish post-attack backdoor persistence as an important security concern in federated learning.

Beyond the model parameters themselves, prior studies have shown that optimization choices and training history can affect attack behavior. Lachnit & Karame (2025) show that optimization hyperparameters can substantially influence backdoor success and persistence. Historical information

has also been studied from an adversarial perspective: Özfatura et al. (2024) examines how Byzantine attackers can exploit training history against centered clipping. Panigrahi et al. (Panigrahi et al., 2023) use momentum within malicious clients, separately maintaining SGD momentum over benign and backdoor data and combining them to construct smoother malicious updates. In their setting, momentum is part of the attacker's local optimization, while the server performs FedAvg.

Optimizer state effect. Optimizer state can itself encode information that affects future training behavior. Xu (2026) study subliminal trait transfer through optimizer-state interventions and show that differences in optimizer moments can leave current parameters and outputs unchanged while producing different behaviors during subsequent training. Optimizer buffers are also important when resuming training from checkpoints (Inkawhich, 2025). In federated learning, methods such as FedAvgM and adaptive federated optimizers explicitly maintain server-side state that accumulates information from previous rounds (Hsu et al., 2019; Reddi et al., 2021). These observations motivate treating the server optimizer state, rather than only the current model parameters, as part of the training state that may influence subsequent model behavior.

However, it remains unclear whether such persistent server-side optimizer state contributes to residual backdoor risk after malicious clients leave, or whether an attacker can intentionally shape this hidden state using only client-side submissions. We will answer the two questions in this work.

3 Dependency of Backdoor Attack on Optimizer State

We first introduce federated learning and backdoor attacks. We then investigate the effect of optimizer state through researcher-controlled interventions, establishing its causal role in subsequent backdoor behavior.

3.1 Federated Learning and Backdoor Attack

Federated Learning. Federated learning (FL) enables multiple clients to collaboratively train a global model without sharing their local data. At each round t, the parameter server (PS) distributes the global model θ_t to selected clients. Each client performs local training on its private dataset and returns an update to the server. The server aggregates the received updates to update the global model. We use a_t to represent the aggregated model updates.

Backdoor Attacks. Malicious clients poison their local training to embed a targeted behavior into the global model while preserving benign performance. They submit poisoned updates to the server.

Federated Optimization with Momentum. Under FedAvgM, the server maintains momentum, and the momentum evolves as

$$m_t = \beta m_{t-1} + a_t, \tag{1}$$

where m_t denotes the momentum, β is the server momentum coefficient, and a_t represents the aggregated model updates. The global model θ is updated by $\theta_{t+1} = \theta_t - \eta m_t$ where η represents the learning rate.

3.2 Characterizing Optimizer-State Risk via Causal Audit

To analyze the backdoor effects of optimizer state, we propose a causal state-auditing framework that isolates the contribution of optimizer state from other factors.

Isolating the Effect of Optimizer State. Let $S_t = (\theta_t, m_t, z_t)$ denote the training state, where θ_t is the model, m_t is the server momentum, and z_t contains the remaining training state. We place the audit boundary immediately before the next server update, after benign client updates have been computed and cached. At this boundary, we hold fixed participating clients and their updates. We explore the server momentum m_t while keeping other quantities fixed, and then continue with benign training. The first server update isolates the effect of momentum under identical model and client updates, while later updates capture its downstream effect on the subsequent training trajectory.

Optimizer-State Intervention. As shown in Figure 1, we construct three branches from the same attacked model checkpoint: *preserve* (P), which retains the server momentum; *reset* (R), which

sets the momentum to zero; and *clean-donor* (D), which replaces it with momentum from a clean run matched by lineage, defense, and training round. All branches retain the same attacked model weights and subsequently undergo benign training. Under FedAvgM (Hsu et al., 2019), the updates are calculated by $\theta_{t+1} = \theta_t - \eta m_t$, yielding

$$\theta^P_{t+1} - \theta^D_{t+1} = \eta(m^D_t - m^P_t). \tag{2}$$

Thus, the first-step difference is induced by the intervened momentum state.

Measuring the State Effect. Let H0 denote the shared model before intervention and Hh the model after h benign server updates. We measure backdoor risk using attack success rate (ASR) and quantify the effect of optimizer state as $\Delta^{P-D}_h = \mathrm{ASR}^P_h - \mathrm{ASR}^D_h$ and $\Delta^{P-R}_h = \mathrm{ASR}^P_h - \mathrm{ASR}^R_h$. At H1, all branches use the same cached client updates, so the comparison measures the immediate effect of optimizer state intervention. For later updates, we consider two continuation protocols. In the *closed-loop* setting, client updates are recomputed from each branch's evolving model, capturing the downstream training trajectory induced by the intervention. In the *open-loop* setting, all branches replay the same sequence of benign aggregates, controlling for differences in subsequent client updates.

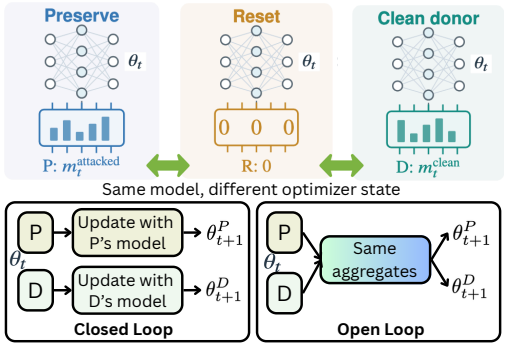

Figure 1: Optimizer state causal auditing.

3.3 Effects of Optimizer Status.

We perform the state causal auditing by using ResNet-18 (He et al., 2016) trained on the CIFAR-10 (Krizhevsky & Hinton, 2009) dataset. The server is implemented with the aggregation algorithm FedAvgM (Hsu et al., 2019) with momentum 0.9. We specifically investigate how optimizer momentum impacts the backdoor effectiveness. The closed-loop setting is selected as a demonstration. Detailed experimental settings are shared in Sec 4.1.

P-D (Preserved momentum vs. clean momentum). The average preserve-minus-donor ASR difference increases from 2.30% at H1 to 6.48% at H3 and 8.82% at H5 (Table 1), while the absolute clean-accuracy difference at H5 is only 0.04%. These results show that replacing attacked momentum with matched clean momentum substantially changes future backdoor behavior while having negligible impact on clean utility. **Open Loop vs. closed loop:** We observe a similar trend under open-loop continuation, where all branches replay the same sequence of benign aggregates and the P-D ASR difference increases from 2.30% at H1 to 9.41% at H5. This result shows that the divergence does not require differences in subsequent benign client updates. The intervened optimizer state alone is sufficient to induce distinct backdoor trajectories under an identical sequence of aggregates. Unless otherwise specified, we use the closed-loop setting in subsequent experiments to capture the natural evolution of each branch during continued training.

Table 1: **ASR Difference Δ.** The last columns report the clean-accuracy difference.

Continuation	H1 ΔASR	H3 ΔASR	H5 ΔASR	H5 $\lvert\Delta\text{Acc}\rvert$
Closed loop	2.296	6.476	8.819	0.040
Open loop	2.296	6.646	9.409	0.152

Moreover, the growing P-D gap from H1 to H5 reveals an interesting contrast: the behavioral divergence becomes more pronounced even as the initial momentum difference decays over time. To understand this phenomenon, we analyze how a transient difference in server momentum propagates into the subsequent model trajectory by the following proposition.

Proposition 1 *(Persistent Parameter Displacement from Decaying Momentum).* Consider two FedAvgM trajectories initialized with identical model parameters, $\Theta^P_0 = \Theta^D_0$, but different server

momentum, $m_0^P \neq m_0^D$. If both trajectories subsequently receive identical aggregated updates, then after h server updates,

$$m_h^P - m_h^D = \beta^h (m_0^P - m_0^D), \tag{3}$$

while

$$\Theta_h^P - \Theta_h^D = \eta_s \frac{\beta(1-\beta^h)}{1-\beta} (m_0^P - m_0^D). \tag{4}$$

Therefore, as $h \to \infty$, the momentum difference vanishes geometrically, whereas the induced parameter displacement generally converges to the nonzero limit. Thus, a transient difference in optimizer state can leave a persistent effect on the model parameters even after the state difference itself has largely decayed. This result does not imply that the ASR gap must grow with the parameter displacement, since ASR is a nonlinear behavioral metric. Rather, it establishes a mechanism by which a decaying optimizer-state difference can induce a persistent difference in the subsequent model trajectory, providing a parameter-space explanation for the sustained behavioral divergence observed empirically.

P-R (preserved momentum vs. zero momentum). We measure the effect of retaining momentum relative to resetting it to zero. The anchor model at H0 is either a clean model or a malicious model. Figure 2 reports the ASR difference at H5, five benign rounds. The orange bars show the variation across lineages if the anchor model is malicious while the grey bars are for clean anchor models. The preserve-minus-reset ASR ranges from 4.98% to 15.64% for attacked anchors, but only from -0.18% to 0.71% for clean anchors. Thus, retaining momentum has a substantial effect following attacked training, while having little effect following clean training. Because the attacked and clean anchors have different model weights, this comparison serves as a matched-lineage diagnostic of attack-history dependence rather than a controlled intervention that holds the model state fixed across histories.

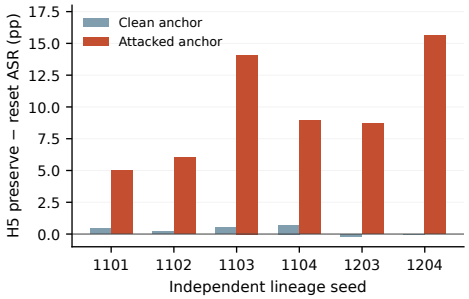

Figure 2: Preserve-minus-reset ASR differences at H5.

3.4 DynaPhantom: From State Dependence to Practical Exploitation

The state causal audit demonstrates that server optimizer state can substantially affect future backdoor risk under researcher-controlled interventions. However, a realistic attacker cannot directly access or modify the server momentum. It can only influence the server state indirectly through its submitted updates. This raises a practical question: can an attacker exploit the identified state dependence in a realistic scenario? To investigate this question, we propose a history-aware attack, DynaPhantom, that strategically conditions malicious updates on prior attack history to shape the evolving server state. The name reflects this design: DynaPhantom dynamically exploits attack history to leave a "phantom" influence in the optimizer state that continues to shape the model trajectory even after malicious participation ends.

Threat Model. We assume a practical attacker that cannot directly observe or modify the server optimizer state. Instead, it can influence the training process only through its malicious client submissions. At each learning round, the attacker has access to its current local information, the observable global model, and its own submissions from previous rounds. Our goal is therefore to determine whether the attacker can exploit its own submissions to increase future backdoor risk without direct access to the server optimizer state.

History-aware Attack. To investigate this question, we develop DynaPhantom, a history-aware attack that uses a summary of its previous submissions to guide the direction of the current malicious update. The motivation is directly from the recursive nature of server momentum. Therefore the server momentum contains a decayed accumulation of updates from previous rounds. Although an attacker cannot observe the complete m_t, its own previous submissions constitute the known portion of the update history through which it has influenced this hidden state. DynaPhantom therefore does not attempt to estimate or reconstruct the server momentum. Instead, it uses its own

submission history to coordinate the attacker-controlled contributions that are repeatedly injected into the momentum recursion. The submitted local update $\hat{u}_t$ is designed as:

$$\hat{u}_t = \|u_t\| \frac{(1-\lambda)u_t + \lambda \hat{u}_{t-1}}{\|(1-\lambda)u_t + \lambda \hat{u}_{t-1}\|}. \tag{5}$$

where u_t is the malicious model update at the current learning round, $0 < \lambda < 1$ is a controllable coefficient that controls the ratio of historical models and the current model, and $\|a\|$ calculates the L-2 norm of vector a. Note that we define $\hat{u}_0$ as the original malicious update u_0 since no history has been collected at time step 0. We further normalize the calculated update by its original norm to avoid an abnormal norm. Here, the current component allows the attack to adapt to the evolving global model, while the historical component promotes temporal directional coherence across malicious submissions. Conditioning on historical updates encourages successive malicious updates to reinforce rather than counteract the attacker's previous contributions. Thus, DYNAPHANTOM can consistently shape the portion of its evolution that the attacker can influence through its own submissions.

Late-Stage State Shaping. DYNAPHANTOM further exploits the temporal nature of optimizer state through a late-stage schedule. Under the same total attack budget, it allocates more attack strength toward the final portion of the malicious participation window. Earlier malicious contributions undergo more subsequent aggregation and benign training before the attacker leaves, whereas later contributions experience fewer opportunities to be diluted before the post-attack state is formed. Combining temporal coherence with late-stage allocation therefore leaves a stronger attacker-induced component in the training state. Together, these mechanisms turn the state dependence revealed by our audit into a practical attack strategy.

4 EVALUATION AND FINDINGS

Our experiments address three research questions: **RQ1:** How does server momentum causally affect future backdoor behavior over time? **RQ2:** How much additional backdoor persistence does history-aware conditioning provide beyond matched baselines? **RQ3:** What are the security implications of the optimizer-state effect? We further examine whether these findings extend beyond image classification through a case study on language models.

4.1 EXPERIMENTAL SETUP

Training and evaluation. We use CIFAR-10 (Krizhevsky & Hinton, 2009) and ResNet-18 (He et al., 2016) with base width 32 and eight-group GroupNorm (Wu & He, 2018). For the training set, we reserve a class-balanced validation set of 2,500 images, with 250 images per class, and distribute the remaining 47,500 training images across 20 clients using a Dirichlet partition with concentration 0.5. Each run begins with 300 pretraining rounds, sampling 10 clients per round; subsequent attack and benign-continuation rounds use all 20 clients. We use FedAvgM (Hsu et al., 2019) with server momentum 0.9. We evaluate our proposed backdoor attack, DYNAPHANTOM, against Neurotoxin (Zhang et al., 2022), a local-proxy adaptation of Neurotoxin, under a common FedAvgM setting with matched training conditions and attack budgets. Neurotoxin explicitly targets backdoor persistence after malicious participation ends. Clean accuracy is measured on all 10,000 unmodified test images. Attack success rate (ASR) is the fraction of trigger-modified images classified as the target class, airplane, among images whose true labels differ from that class. Excluding the 1,000 airplane images gives 9,000 test images for ASR evaluation. For each independently pretrained run, we hold the anchor model weights fixed and compare preserved server momentum with momentum from a matched clean run.

Metrics. We evaluate backdoor effectiveness using the standard attack success rate (ASR), defined as the fraction of attack samples misclassified into the target label. To isolate the effect of optimizer state, we focus on the paired ASR difference rather than the absolute ASR of each branch. Specifically, at horizon h, we compute $\Delta_h^{P-D} = \mathrm{ASR}_h^P - \mathrm{ASR}_h^D$ and $\Delta_h^{P-R} = \mathrm{ASR}_h^P - \mathrm{ASR}_h^R$. A larger positive difference indicates a stronger residual backdoor effect attributable to the retained optimizer state. We also use clean accuracy to report the model performance on clean data.

4.2 Evaluation Results

Optimizer state leads to diverging future backdoor risk. Figure 3 shows the continuation behavior of different optimizer state interventions. Recall that H0 denotes the shared model before intervention and Hh the model after h benign server updates. Starting from the same H0 model, the preserve, clean-donor, and reset branches exhibit increasingly different backdoor behavior during subsequent benign training(3 (a)). At H5, preserving the attacked momentum results in an average ASR of 65.35%, compared with 35.42% after replacement with clean momentum. The preserve-minus-donor difference grows consistently across all three lineages and reaches an average of 29.93% at H5 (3 (b)). In contrast, clean-accuracy trajectories remain close across the three state interventions, with H5 accuracy ranging from 79.43% to 79.71% (3 (c)). These results show that different optimizer states can lead the same attacked model to substantially different future backdoor behavior while preserving similar clean utility.

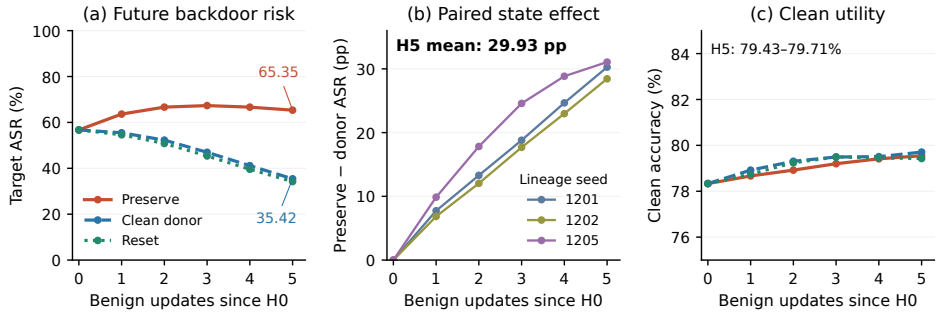

Figure 3: **Optimizer state effects.** The ASR gap grows while clean-accuracy trajectories remain close.

Table 2 compares DynaPhantom with five baselines and ablation variants to assess the contribution of history conditioning. *Current-only* uses only information from the current round. *Raw dual history*, *Reconstructed momentum*, and *Handcrafted CMI* incorporate historical information using raw update histories, estimated server momentum, and similarity-weighted history, respectively. Specifically, *Handcrafted CMI* uses cosine similarity to weight historical information, such that history more similar to the current update contributes more strongly. DynaPhantom conditions submissions on an exponential moving average of its own past submissions. DynaPhantom achieves the highest ASR among all evaluated methods from H1 to H5. At H5, its ASR of 88.43% exceeds *Current-only* by approximately 27 percentage points, while the absolute clean-accuracy difference is at most 0.16 percentage points. In the 'ASR-D' column, we replace the attacked momentum with matched clean momentum. We observe that benign momentum will decrease the ASR dramatically compared with malicious momentum. We also observe two ASR trajectories from H1 to H5: some methods first increase and then decrease, while others decrease directly. A possible explanation is that retained momentum may initially reinforce the backdoor direction in the former case, whereas subsequent benign updates gradually weaken this effect.

Table 2: **ASR (%) of DynaPhantom and other Baselines.**

Method	H1	H2	H3	H4	H5	ASR-D (H5)	Accuracy (H5)
Current-only	56.98	61.21	62.56	62.22	61.04	32.36	79.56
Raw dual history	59.48	62.83	63.30	62.37	60.80	31.67	79.64
Reconstructed momentum	52.15	56.89	59.14	59.72	58.99	30.21	78.95
Handcrafted CMI	58.41	62.56	63.97	63.59	62.34	33.38	79.59
Neurotoxin (Zhang et al., 2022)	73.70	71.79	71.67	71.32	70.32	32.17	73.95
DynaPhantom	90.65	89.36	88.79	88.53	88.43	35.42	79.55

4.3 Security Implications for Checkpoint Recovery

The preceding state interventions demonstrate that optimizer state can affect future backdoor behavior. We next examine the practical implication for checkpoint recovery. In particular, when training

resumes from the same saved model weights, does the choice of optimizer-state recovery policy affect the subsequent security trajectory? We compare three recovery policies from the same attacked checkpoint: *weights-only*, which restores the model weights and resets server momentum to zero; *full*, which restores both the model weights and the saved server momentum; and *reconstructed*, which restores the same weights but reconstructs momentum from adjacent model snapshots. Thus, all three policies resume from identical model weights but differ in how the optimizer state is recovered.

Although all three policies restore identical model weights, they produce different continuation behavior. Compared with weights-only recovery, full recovery results in an average ASR difference of 3.78% at H1 and 12.16% at H5, while the mean absolute H5 clean-accuracy difference is only 0.19%. Weights-only recovery reduces future backdoor risk but does not eliminate it, with a mean H5 ASR of 11.39%. In contrast, reconstructed recovery closely reproduces full recovery. Across the three lineages, the reconstructed momentum has a relative L2 error of only 1.20×10^{-5}–1.25×10^{-5}, and the maximum absolute Full–Reconstructed ASR difference over H1–H5 is 0.18%. These results show that restoring the optimizer state, rather than the model weights alone, is important for reproducing the future security trajectory.

Table 3: ASR (%) at Round 5 after Checkpoint Restoration.

Lineage	Full	Weights-only	Reconstructed
1102	13.78	6.31	13.73
1103	33.07	16.80	33.24
1104	23.82	11.07	23.96
Mean	23.56	11.39	23.64

4.4 Case Study: Effect of Optimizer State in Federated Language-Model Adaptation

Our evaluation on image classification demonstrates that optimizer state can causally alter future backdoor risk. We next examine whether this effect extends to federated language-model adaptation, which differs substantially from image classification in model architecture, task structure, and adaptation mechanism. Specifically, we study Llama-2-7B (Touvron et al., 2023) with FFA-LoRA adaptation (Sun et al., 2024) on SST-2 (Socher et al., 2013), a widely used binary sentiment classification dataset consisting of movie-review sentences labeled as positive or negative. We evaluate ASR on a fixed set of 138 malicious inputs and clean accuracy on 436 benign inputs. We otherwise follow the federated learning setup described in Section 4, including the use of FedAvgM for server aggregation.

Starting from an existing H1 preserve branch (P) and clean-donor branch (D), we continue both for 19 additional benign rounds. Later client updates are recomputed from each branch's evolving model, so the experiment follows their respective closed-loop continuation trajectories. Figure 4 shows that the P-D risk gap increases substantially during benign continuation, from 24.64% at H1 to 40.58% at H5, 61.59% at H10, and 85.99% at H20. The widening gap reflects different recovery dynamics rather than increasing risk in P: P remains at 138/138 target responses from H3 through H20, whereas D progressively recovers.

To further illustrate the individual behavior of the P and D branches, Table 4 reports their outcomes at the 20th round after momentum intervention across three independent schedules. The effect is highly consistent: the P–D ASR gap ranges from 84.78% to 87.68% across the three schedules. In contrast, their clean behavior remains closely matched. At H5, P and D produce identical predictions on all 436 clean inputs across all three schedules. Even at H20, their clean predictions differ on only one input for S01 and S02 and remain identical for S03, corresponding to a mean signed clean-accuracy difference of only 0.15%. These results show that the P and D branches can follow substantially different backdoor trajectories while remaining nearly indistinguishable in their clean predictions.

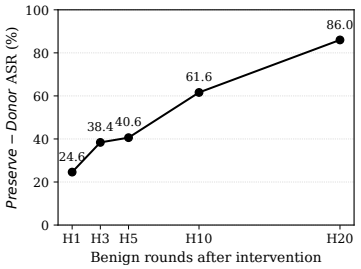

Figure 4: ASR gap between preserved and clean-donor branches.

Table 4: **ASR and clean accuracy for three schedules at Round 20.**

Schedule	ASR-P	ASR-D	Gap (%)	Clean Accuracy-P	Clean Accuracy-D	Gap
S01	138/138	20/138	85.51	418/436	417/436	1/436
S02	138/138	17/138	87.68	418/436	417/436	1/436
S03	138/138	21/138	84.78	417/436	417/436	0/436

4.5 Potential Defense

In this work, we identify that optimizer state can retain the influence of malicious updates and contribute to backdoor persistence in federated learning. This finding suggests a straightforward defense: once malicious model updates are identified, the server should reset the corresponding optimizer state to prevent their residual influence. Existing defenses (Jia et al., 2025) (Rieger et al., 2024) Kabir et al. (2024) provide various mechanisms for detecting malicious model updates. Our finding suggests an additional step for these defenses: resetting the optimizer state after malicious updates are detected.

5 Discussion, Statements, and Conclusion

Discussion. Our findings highlight several directions for future research. First, defenses against federated backdoors should consider optimizer state in addition to model parameters. Since malicious influence can persist in server momentum and affect future training, monitoring or sanitizing optimizer state may complement existing update- and model-level defenses. However, simply resetting momentum does not guarantee backdoor removal, as malicious behavior may already be encoded in the model parameters. An important direction is therefore to jointly mitigate backdoor influence in both model parameters and optimizer state. Second, optimizer state should be considered when storing and restoring training checkpoints. Preserving state enables faithful training continuation but may also preserve hidden backdoor risk, whereas weights-only restoration can alter the subsequent security trajectory. Secure checkpointing should therefore account for both model parameters and optimizer state. Finally, our experiments study finite continuation horizons. Future work can investigate when state-induced security differences vanish, persist, or amplify over longer training periods and under changing client populations, data distributions, and optimization algorithms.

Ethics Statement. This work studies residual backdoor risk to improve the security evaluation of federated training and checkpoint recovery. While our findings may also inform adversaries about backdoor persistence, our study focuses on understanding such risks under clearly defined threat models. The state interventions considered in this work are research tools for causal analysis rather than attack procedures.

Reproducibility Statement. We will provide our code as supplementary material and plan to release it publicly on GitHub upon acceptance. The release will include experiment configurations, environment specifications, analysis scripts, data-split and client-partition indices, and recorded client-selection schedules needed to reproduce the reported results.

Statement on the Use of Large Language Models. Large language models were used only to assist with writing, editing, and formatting. They were not used to develop the core methods, conduct experiments, analyze results, or generate scientific claims.

Conclusion. We identify a previously overlooked security risk in federated optimization: malicious training history can continue to influence backdoor behavior through retained server momentum even after the attacker leaves. Through controlled state interventions, we establish this causal effect by showing that identical model weights can exhibit substantially different future backdoor behavior solely due to differences in optimizer state. We further introduce DynaPhantom, a history-aware attack that exploits this state dependence through malicious submission history without requiring direct access to the server optimizer state. Our experiments show that DynaPhantom substantially strengthens post-attack backdoor persistence. Moreover, our checkpoint recovery experiments reveal that restoring identical model weights does not guarantee the same subsequent security trajectory. Together, these findings show that model weights and clean performance alone are insufficient to characterize residual backdoor risk when training resumes. Server optimizer state should therefore be treated as security-relevant state in both backdoor evaluation and checkpoint recovery.

REFERENCES

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2938–2948. PMLR, 2020. URL https://proceedings.mlr.press/v108/bagdasaryan20a.html.

Yijie Bai, Yanjiao Chen, Hanlei Zhang, Wenyuan Xu, Haiqin Weng, and Dou Goodman. {VILLAIN}: Backdoor attacks against vertical split learning. In *Proc. USENIX Security Symp.*, pp. 2743–2760, 2023.

Yuanpu Cao, Bochuan Cao, and Jinghui Chen. Stealthy and persistent unalignment on large language models via backdoor injections. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4920–4935, 2024.

Yanbo Dai and Songze Li. Chameleon: Adapting to peer images for planting durable backdoors in federated learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 6712–6725. PMLR, 2023. URL https://proceedings.mlr.press/v202/dai23a.html.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019. doi: 10.1109/ACCESS.2019.2909068.

Zhen Guo, Abhinav Kumar, and Reza Tourani. Persistent backdoor attacks in continual learning. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 6379–6397, 2025.

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*, pp. 770–778, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019. URL https://arxiv.org/abs/1909.06335.

Matthew Inkawhich. Saving and loading models. PyTorch Tutorials, 2025. URL https://docs.pytorch.org/tutorials/beginner/saving_loading_models.html. Created August 29, 2018; updated June 26, 2025. Accessed September 22, 2026.

Yuqi Jia, Minghong Fang, Hongbin Liu, Jinghuai Zhang, and Neil Gong. Tracing back the malicious clients in poisoning attacks to federated learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3882. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a924b7178e5975dfed1de235f0b72973-Abstract-Conference.html.

Ehsanul Kabir, Zeyu Song, Md Rafi Ur Rashid, and Shagufta Mehnaz. FLShield: A validation based federated learning framework to defend against poisoning attacks. In *2024 IEEE Symposium on Security and Privacy*, 2024. doi: 10.1109/SP54263.2024.00141.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Simon Lachnit and Ghassan Karame. On hyperparameters and backdoor-resistance in horizontal federated learning. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1919–1933. ACM, 2025. doi: 10.1145/3719027.3765211. URL https://arxiv.org/abs/2509.05192.

Tao Liu, Yuhang Zhang, Zhu Feng, Zhiqin Yang, Chen Xu, Dapeng Man, and Wu Yang. Beyond traditional threats: A persistent backdoor attack on federated learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(19):21359–21367, 2024. doi: 10.1609/aaai.v38i19.30131. URL https://ojs.aaai.org/index.php/AAAI/article/view/30131.

Jiayi Luo, Qingyun Sun, Lingjuan Lyu, Ziwei Zhang, Haonan Yuan, Xingcheng Fu, and Jianxin Li. Towards effective, stealthy, and persistent backdoor attacks targeting graph foundation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24142–24150, 2026.

Kerem Özfatura, Emre Özfatura, Alptekin Küpçü, and Deniz Gündüz. Byzantines can also learn from history: Fall of centered clipping in federated learning. *IEEE Transactions on Information Forensics and Security*, 19:2010–2022, 2024. doi: 10.1109/TIFS.2023.3345171. URL https://arxiv.org/abs/2208.09894.

Satwik Panigrahi, Nader Bouacida, and Prasant Mohapatra. Double momentum backdoor attack in federated learning. *Journal of Student Research*, 12(1):3644, 2023.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2003.00295.

Phillip Rieger, Torsten Krauß, Markus Miettinen, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. CrowdGuard: Federated backdoor detection in federated learning. In *Network and Distributed System Security Symposium*, 2024. doi: 10.14722/ndss.2024.23233.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. Association for Computational Linguistics, 2013. URL https://aclanthology.org/D13-1170/.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving LoRA in privacy-preserving federated learning. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4e243e95c913b367775d71d7182b99d9-Abstract-Conference.html.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision*, pp. 3–19, 2018. URL https://openaccess.thecvf.com/content_ECCV_2018/html/Yuxin_Wu_Group_Normalization_ECCV_2018_paper.html.

Yueqi Xie, Minghong Fang, and Neil Zhenqiang Gong. Model poisoning attacks to federated learning via multi-round consistency. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 15454–15463, 2025.

Qinyang Xu. Stored in optimizer state, valued by later training: A causal account of subliminal trait transfer. arXiv preprint arXiv:2608.20442, 2026. URL https://arxiv.org/abs/2608.20442.

Fangchao Yu, Bo Zeng, Kai Zhao, Zhi Pang, and Lina Wang. Chronic poisoning: Backdoor attack against split learning. In *Proc. AAAI Conf. Artif. Intell.*, volume 38, pp. 16531–16538, 2024.

Zhengming Zhang, Ashwinee Panda, Linyue Song, Yaoqing Yang, Michael W. Mahoney, Joseph E. Gonzalez, Kannan Ramchandran, and Prateek Mittal. Neurotoxin: Durable backdoors in federated learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 26429–26446. PMLR, 2022. URL https://proceedings.mlr.press/v162/zhang22w/zhang22w.pdf.

Wenwei Zhao, Yuanzhe Peng, Xiaowen Li, Jie Xu, Yao Liu, and Zhuo Lu. Malicious forgetting: Backdoor injection in active federated unlearning and countermeasure design. In *IEEE INFOCOM 2026 - IEEE Conference on Computer Communications*, pp. 1–10, 2026. doi: 10.1109/INFOCOM59046.2026.11571403.