Chronic Poisoning: Backdoor Attack against Split Learning

Fangchao Yu, Bo Zeng, Kai Zhao, Zhi Pang, Lina Wang*

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University
{fangchao, bobozen, kizhao, zhipang, lnwang}@whu.edu.cn

Abstract

Split learning is a computing resource-friendly distributed learning framework that protects client training data by splitting the model between the client and server. Previous work has proved that split learning faces a severe risk of privacy leakage, as a malicious server can recover the client's private data by hijacking the training process. In this paper, we first explore the vulnerability of split learning to server-side backdoor attacks, where our goal is to compromise the model's integrity. Since the server-side attacker cannot access the training data and client model in split learning, the traditional poisoning-based backdoor attack methods are no longer applicable. Therefore, constructing backdoor attacks in split learning poses significant challenges. Our strategy involves the attacker establishing a shadow model on the server side that can encode backdoor samples and guiding the client model to learn from this model during the training process, thereby enabling the client to acquire the same capability. Based on these insights, we propose a three-stage backdoor attack framework named SFI. Our attack framework minimizes assumptions about the attacker's background knowledge and ensures that the attack process remains imperceptible to the client. We implement SFI on various benchmark datasets, and extensive experimental results demonstrate its effectiveness and generality. For example, success rates of our attack on MNIST, Fashion, and CIFAR10 datasets all exceed 90%, with limited impact on the main task.

Introduction

Split learning is a computing resource-friendly distributed learning framework, considered as a supplement to federated learning (Gao et al. 2020b; Kairouz et al. 2021; Li et al. 2020). In split learning, a complete neural network model is divided into two parts: the client model and the server model, which are held by the client and server, respectively. Typically, the client only needs to perform lightweight calculations, and offloads the rest laden computations to the server. During the forward propagation phase, the client model encodes the local data into feature vectors, serving as inputs to the server model. In the backpropagation process, the server sends gradient information to the client to facilitate updates of the client model. Since the server cannot directly access the client model and training data, split learning is regarded as a secure and privacy-preserving training strategy (Vepakomma et al. 2018; Singh et al. 2019).

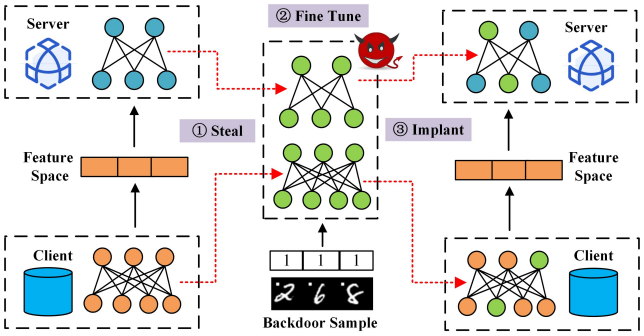


Figure 1: Overview of the backdoor attack framework against split learning.

However, as research progresses, the security risks associated with split learning have gradually come to light (Zhang et al. 2023). Some recent work have presented a series of attacks targeting private client data, including data reconstruction attacks (Pasquini, Ateniese, and Bernaschi 2021) and label inference attacks (Fu et al. 2022). In this paper, we first demonstrate the vulnerability of split learning to backdoor attacks (Li et al. 2022; Gao et al. 2020a). We believe that the malicious server has both the motivation and capability to implant backdoors in the model, thereby achieving specific objectives.

In split learning, constructing backdoor attacks from the server side poses a significant challenge. Traditional approaches of constructing backdoor attacks rely on controlling the training data, wherein the attacker injects backdoor samples into the training data to confer the model with the ability to encode the backdoor samples (Bagdasaryan and Shmatikov 2020; Nguyen and Tran 2021; Wenger et al. 2021; Bagdasaryan et al. 2020; Zhang et al. 2022). However, in the context of split learning, the server cannot access or interact with the training data, and the model responsible for encoding the input data is under the client's control. Consequently, the conventional strategies for backdoor attacks become inapplicable. We contend that the pivotal breakthrough for constructing server-side backdoor at-

*Correspoonding Author.

tacks lies in fully leveraging the absolute control exerted by the server over the training process. An practical strategy is that the attacker can manipulate the optimization direction of gradients, thereby enabling the client models to indirectly acquire the capability of encoding the backdoor samples.

Based on this insight, we introduce a three-stage backdoor attack framework known as SFI (**S**teal, **F**inetune, and **I**mplant), as shown in Figure 1. Firstly, the client and server start the normal training process. At the same time, we leverage the main task model (server model) and a limited amount of data accessible to the attacker, which includes the backdoor samples, to train a shadow model. The shadow model can encode the backdoor samples and serves the purpose of guiding the client model towards acquiring the same capability. Next, we proceed with finetuning both the shadow and main task models to enhance their performance in terms of the main task and backdoor attack. Finally, we regard the client models as generators and construct a knowledge transfer system on the server side utilizing a discriminator and the main task model to jointly guide the client model learning from the shadow model.

Through the three stages mentioned above, the attacker can successfully implant backdoor in the split learning model. The main contributions of this paper can be summarized as follows:

- We propose a novel backdoor attack framework against split learning. Our attack exhibits several advantages: (1) minimizing background knowledge assumptions, the attacker only needs hundreds or thousands of data samples to successfully implement attacks. (2) The attack process is transparent to the client, and the impact on the main task is limited, making it difficult for clients to detect malicious behavior.
- We design a strategy to improve the performance of backdoor attacks. The attacker can introduce an auxiliary model on the server side to enhance the sensitivity of the shadow model to backdoor samples, thereby alleviating the sensitivity loss in the backdoor implantation stage.
- We comprehensively evaluate the effectiveness and generality of the SFI backdoor attack framework. For example, for the MNIST, Fashion, and CIFAR10 datasets, the attacker only requires 500, 750, and 2500 shadow dataset samples to achieve backdoor attack accuracy of 92.8%, 97.4%, and 91.8%, respectively. The introduction of the auxiliary model improves the accuracy of backdoor attacks on the MNIST dataset by 8%, without a negative impact on the performance of the main task.

Related Work

In this section, we will review the research progress of split learning security, and focus on the data reconstruction attacks and label inference attacks.

Data reconstruction attack. In a data reconstruction attack, the attacker is on the server side and aims to recover the original training data from the feature vector uploaded by the clients. We categorize existing data reconstruction attacks in split learning into two types: (1) the passive attack, where the attacker does not disrupt the normal split learning training process and instead leverages intermediate information from the training process to construct an attack model. Examples of such attack include Unsplit (Erdoğan, Küpçü, and Çiçek 2022) and PCAT (Gao and Zhang 2023). (2) the active attack, where the attacker manipulates the normal split learning training process to acquire the ability to reconstruct client data. FSHA (Pasquini, Ateniese, and Bernaschi 2021) is an example of such an attack. The active attack is more flexible than the passive attack but is more easily detected by the client. Erdogan et al. (Erdogan, Küpçü, and Cicek 2022) argue that the weight update direction of the clients in FSHA is independent of the main task. Thus, clients can submit a small amount of erroneous data during the training process to observe the variations in gradient information returned by the server, enabling them to detect any malicious behavior on the server side. Fu et al. (Fu et al. 2023) point out that the client model can essentially be viewed as the encoder of an autoencoder structure in FSHA, which is insensitive to the sample labels. Thus, the client can detect the presence of malicious attack by examining the intrinsic differences between the expected model's gradients and the malicious model's gradients.

Label inference attacks. Label inference attacks involve split learning participants who aim to steal the label information of training data, incentivizing both the server and the client to engage in such attacks. Li et al. (Li et al. 2021) present the first method to measure label privacy in split learning and develop two simple and practical attacks to recover label information accurately. However, these strategies are only suitable for binary classification tasks. Fu et al. (Fu et al. 2022) observe that the client model gains strong feature representation capability during training, which the attacker can exploit as a pre-trained model for label inference attacks. Furthermore, adjusting the optimization strategy to enhance the system's dependence on the malicious client model can improve the attack performance. Kariyappa et al. (Kariyappa and Qureshi 2022) adjust the loss function for split learning training tasks and redefine label inference attacks as a supervised learning problem. Liu et al. (Liu and Lyu 2022) research security risks in gradient information and data exchange, proposing a clustering label inference attack with cosine and Euclidean distance measurements, which exhibits high accuracy and adaptability.

Background

In this section, we present the threat model of backdoor attacks and our motivation.

Threat Model

This work aims to design a general and efficient attack framework for implanting backdoors in split learning models. Our attack framework satisfies the following requirements and assumptions:

- The attack process is dominated by a semi-honest server, which lacks knowledge of the client model's information but can obtain limited data samples (similar to the main task training data) from public.

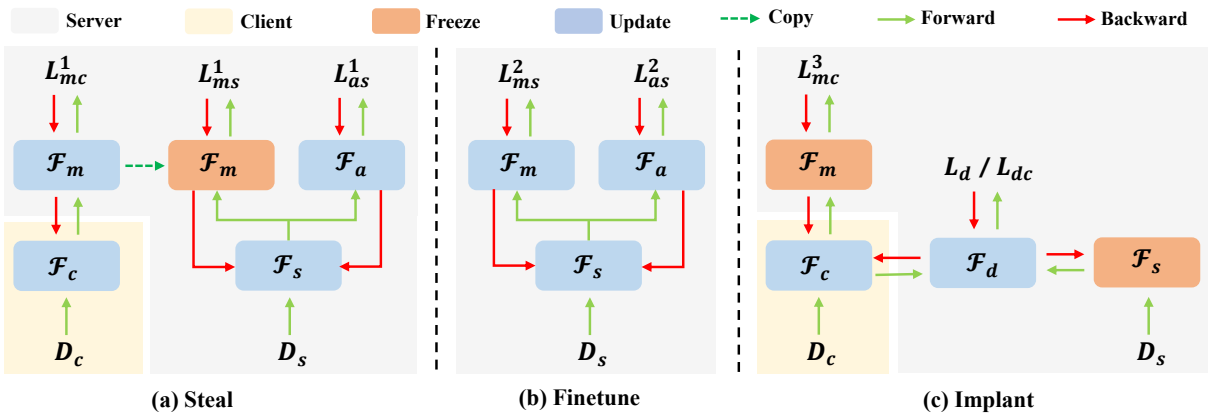


Figure 2: The three stages of SFI. In the Steal stage, the training process involves the simultaneous training of $\mathcal{F}_m$ and $\mathcal{F}_c$, followed by the training of $\mathcal{F}_a$ and $\mathcal{F}_s$ with fixed $\mathcal{F}_m$. In the Finetune stage, the models $\mathcal{F}_m$, $\mathcal{F}_s$, and $\mathcal{F}_a$ are jointly trained and updated. In the Implant stage, we freeze the model parameters of $\mathcal{F}_s$ and $\mathcal{F}_m$, while the main training objects are $\mathcal{F}_c$ and $\mathcal{F}_d$.

- The attack process is transparent to the client. From the client's perspective, the server strictly adheres to the training protocol of split learning, making the attack indistinguishable from honest training.
- To minimize the risk of exposure the attack, the training process should minimize its impact on the main task.
- The attack framework possesses a certain level of generality and can adapt to complex tasks and scenarios (e.g., different datasets and split strategies).

Motivation

As we analyze above, traditional backdoor attack strategies are not applicable for server-side attacker of split learning. On the one hand, the server-side attacker cannot access the training data. On the other hand, the model responsible for encoding the input data is controlled by the client. Therefore, we need to explore some new perspectives.

By delving deeper into the essence of backdoor attacks, we discover that the crucial factor for a successful attack lies in the model's ability to encode backdoor samples. Adding backdoor samples to the training data is the most straightforward and efficient way to acquire this capability. However, is there an indirect way that enables the model to acquire the same capability? Inspired by FSHA, we realize that a model's ability can be acquired not only from the training data but also from other models. Consequently, a spark emerges in our minds: the attacker can build a server-side model capable of encoding backdoor samples and guide the client model to learn from it, thereby indirectly inheriting the same functionality. Considering the server's advantage in absolute control over the optimization direction of the client's gradients, we believe this approach is feasible. Following this idea, we need to address two key challenges:

How to build a shadow model with limited background knowledge? The shadow model is created by the attacker with the capability to encode backdoor samples, guiding the client model in acquiring the ability to encode backdoor samples. Given the strict limitations on the attacker's background knowledge, including limited available data samples and a lack of knowledge about the client model, it is crucial to construct a robust shadow model for the attack's success. On the one hand, we should strive to enhance the sensitivity of the shadow model to backdoor samples and improve the accuracy of the backdoor attack. On the other hand, to minimize the impact on the main task during the attack process, the shadow model's performance on the main task should closely resemble that of the client model.

How to guide the client model to learn from the shadow model? In the normal training process, the optimization direction of the client model is dominated by the main task model on the server. However, we aim to transfer the capability of encoding backdoor samples from the shadow model to the client model. So, we need to build a connection between these two models. We face several challenges: (1) the attacker should minimize the impact on the main task while transferring the backdoor capability, thereby reducing the risk of attack exposure. (2) the approach used by the attacker to control the optimization direction of the client model should be both general and efficient enough to deal with different scenarios and split learning settings.

Apporach

In this section, we provide a detailed description of the SFI framework, as shown in Figure 2. Firstly, we adopt a global perspective to introduce the main components of SFI. Then, we present an in-depth analysis of the three key attack processes. The Steal and Finetune stages aim to establish a robust shadow model, while the Implant stage focuses on guiding the client model to learn from the shadow model. Our code is available from https://github.com/chaoge123456/chronic_poisoning.

Attack Framework

Our attack framework mainly consists of two datasets and five models. $D_c = \{(x_c^i, y_c^i)|i = 1, ..., p\}$ represent the client's dataset, where x_c^i and y_c^i denote feature and label of the data, respectively. $D_s = \{(x_s^i, y_s^i, b_s^i)|i = 1, ..., q\}(q << p)$ represents the shadow dataset, a limited

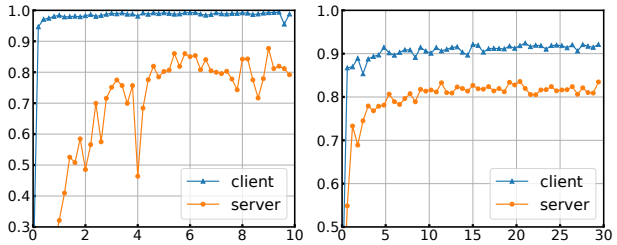
set of data samples obtained by the attacker through public channels. The attacker randomly selects m samples from D_s as the backdoor samples, where a backdoor trigger is inserted into the feature x_s^i, and the corresponding label y_s^i is replaced with target label t. Then, the variable b_s^i can indicate whether a sample in D_s is a backdoor sample, with $b_s^i = 1$ denoting a backdoor sample and $b_s^i = 0$ denoting a clean sample. Typically, during the training process, we divide D_c and D_s into multiple batches, with each batch of data denoted as (X_c, Y_c) and (X_s, Y_s, B_s), respectively. Our attack framework includes the following models:

- client model $\mathcal{F}_c$: $\mathcal{F}_c$ is shared among all clients and takes the local client data features X_c as input while producing the feature vector $Z_c = \mathcal{F}_c(X_c)$ as output.
- main task model $\mathcal{F}_m$: $\mathcal{F}_c$ and $\mathcal{F}_m$ collectively form the core of the split learning task. we assume that the main task is a classification task in this paper. $\mathcal{F}_m$ takes Z_c as input and generates the classification results as output.
- shadow model $\mathcal{F}_s$: $\mathcal{F}_s$ is the object that the client model imitates. $\mathcal{F}_s$ takes X_s as input and produces the feature vector $Z_s = \mathcal{F}_s(X_s)$ as output.
- auxiliary model $\mathcal{F}_a$: $\mathcal{F}_a$ is a binary classification model used to determine whether a training sample is a backdoor sample. $\mathcal{F}_a$ takes Z_s as input and generates the classification results. We believe that $\mathcal{F}_a$ can enhance the sensitivity of $\mathcal{F}_s$ towards backdoor samples to some extent, thereby gaining an advantage in subsequent attacks.
- discriminator $\mathcal{F}_d$: $\mathcal{F}_d$ is crucial in guiding the client model to learn from the shadow model. We consider $\mathcal{F}_c$ as the generator, where Z_s represents true data, and Z_c represents fake data. These three components, along with $\mathcal{F}_d$, form a generative adversarial network system (Creswell et al. 2018).

Steal

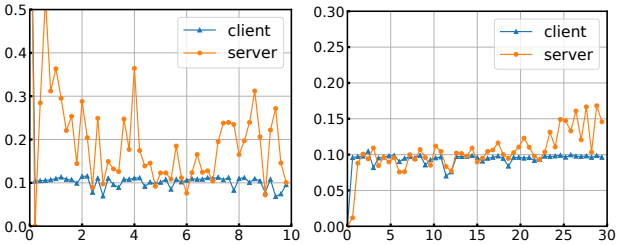
According to our attack strategy, establishing a shadow model is the primary challenge. Due to limited access to background knowledge, the attacker cannot independently construct an effective shadow model. Therefore, the attacker needs to exploit potential training resources as much as possible. Upon reevaluating the split learning framework, we find that the server model $\mathcal{F}_m$ can be sufficiently trained during the normal training process, which implies that $\mathcal{F}_m$ encodes extensive abstract data knowledge (Gao and Zhang 2023). Inspired by this, a natural idea is to utilize $\mathcal{F}_m$ to guide the training of the shadow model $\mathcal{F}_s$, thereby achieving the transfer of both data knowledge and model capabilities. Our strategy is to fix $\mathcal{F}_m$ after each client iteration round and focus on training $\mathcal{F}_s$. The attacker can determine the number of iterations for $\mathcal{F}_s$. This approach allows us to leverage the intermediate information from the normal training process to guide the training of $\mathcal{F}_s$ while alleviating overfitting issues arising from limited training data.

It is worth noting that we introduce an auxiliary model $\mathcal{F}_a$ into the training framework. Our ultimate goal is to transfer the backdoor encoding capability of $\mathcal{F}_s$ to $\mathcal{F}_c$. However, we found that this mode of capability transfer incurs information losses and adversely affects the attack's effectiveness. One feasible solution is to mitigate the information loss in the transfer process by maximizing $\mathcal{F}_s$'s sensitivity to backdoor samples. The most straightforward approach to enhance backdoor sensitivity is to increase the proportion of backdoor samples in the training data. However, this method leads to a decline in the model's performance on the main task, particularly when the attacker's access to training data is limited. We believe that the sensitivity of the model to backdoor samples fundamentally measures its ability to accurately recognize and encode such samples, which implies that we can achieve this goal by increasing the distinguishability between backdoor samples and clean samples. Essentially, the auxiliary model $\mathcal{F}_a$ is a binary classification model aiming to maximize the dissimilarity between clean and backdoor samples. Furthermore, another advantage of this strategy is that it does not impact the performance of the main task.



(a) MNIST dataset (b) Fashion dataset

Figure 3: The main task performance of the cilent ($\mathcal{F}_m$ and $\mathcal{F}_c$) and server ($\mathcal{F}_m$ and $\mathcal{F}_s$) in the Steal stage.



(a) MNIST dataset (b) Fashion dataset

Figure 4: The backdoor attack performance of the cilent ($\mathcal{F}_m$ and $\mathcal{F}_c$) and server ($\mathcal{F}_m$ and $\mathcal{F}_s$) in the Steal stage.

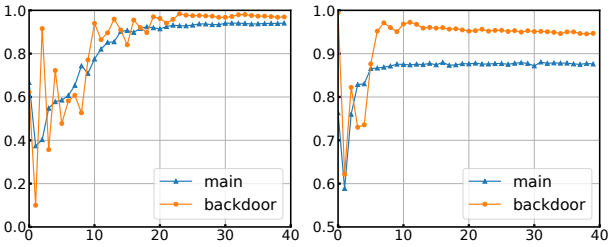
Overall, in the Steal phase, our training process is as follows. Firstly, the client model $\mathcal{F}_c$ and the server model $\mathcal{F}_m$ undergo normal training on the main task, with the loss function ($Z_c = \mathcal{F}_c(X_c)$ and $Z_s = \mathcal{F}_s(X_s)$) :

$$L_{mc}^1 = L(\mathcal{F}_m(Z_c), Y_c) \tag{1}$$

When the training of one client is completed and switches to the next client, the attacker initiates the training of $\mathcal{F}_s$ and $\mathcal{F}_a$ (with $\mathcal{F}_m$ fixed). The loss functions for $\mathcal{F}_s$ and $\mathcal{F}_a$ respectively are defined as follows:

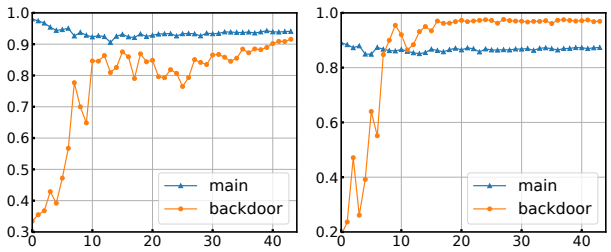
$$L_{ms}^1 = L(\mathcal{F}_m(Z_s), Y_s) \tag{2}$$

$$L_{as}^1 = L(\mathcal{F}_a(Z_s), B_s) \tag{3}$$



(a) MNIST dataset (b) Fashion dataset

Figure 5: The main task and backdoor attack performance of shadow model ($\mathcal{F}_m$ and $\mathcal{F}_s$) in the Finetune stage.



(a) MNIST dataset (b) Fashion dataset

Figure 6: The main task and backdoor attack performance of client model ($\mathcal{F}_m$ and $\mathcal{F}_c$) in the Implant stage.

Finetune

As shown in Figure 3, after the completion of the Steal stage, there still exists a considerable performance gap between $\mathcal{F}_s$ and $\mathcal{F}_c$ on the main task. Furthermore, the performance of $\mathcal{F}_s$ and $\mathcal{F}_c$ in the backdoor attack is also unsatisfactory, as illustrated in Figure 4. Hence, before proceeding with the backdoor implantation, it is imperative to overcome these challenges.

We think the root cause of the abovementioned issues lies in the inadequate alignment between $\mathcal{F}_s$ and $\mathcal{F}_m$. During the first stage, to minimize the impact on the normal training process, we never update $\mathcal{F}_m$ synchronously when training $\mathcal{F}_s$. Therefore, further fine-tuning is necessary to enhance the consistency between $\mathcal{F}_m$ and $\mathcal{F}_s$ during inference. Our strategy involves the attacker training $\mathcal{F}_s$, $\mathcal{F}_m$, and $\mathcal{F}_a$ using D_s, updating their model parameters simultaneously. This process does not require client participation. The loss functions during the training process are as follows:

$$L^2_{ms} = L(\mathcal{F}_m(Z_s), Y_s) \tag{4}$$

$$L^2_{as} = L(\mathcal{F}_a(Z_s), B_s) \tag{5}$$

Generally, after several rounds of finetuning, the model reaches a convergent state, resulting in significant improvements in performance for both the main task and the backdoor attack task. Figure 5 presents the training results of the Finetune stage.

Implant

After the previous two stages, the attacker obtains a well-performing shadow model using limited background knowledge. The next challenge is to guide the client model to learn from the shadow model. During the previous training process, $\mathcal{F}_c$ and $\mathcal{F}_s$ are independently trained. To achieve our objective, we need to introduce a new training mechanism to establish a direct connection between them. The feature space hijacking attack proposed by Pasquini et al. (Pasquini, Ateniese, and Bernaschi 2021) provides us with some inspiration. They establish a loss between the feature spaces Z_c and Z_s as a bridge of $\mathcal{F}_c$ and $\mathcal{F}_s$ and introduce a discriminator to control the gradient updates of $\mathcal{F}_c$. However, The defect of this strategy is that $\mathcal{F}_c$ must abandon the ability accumulated in the previous training process and update parameters with the new loss function as the target, which will lead to a sharp decline in the performance of $\mathcal{F}_c$ on the main task in the early stage of training and increase the risk of exposure to attack behavior.

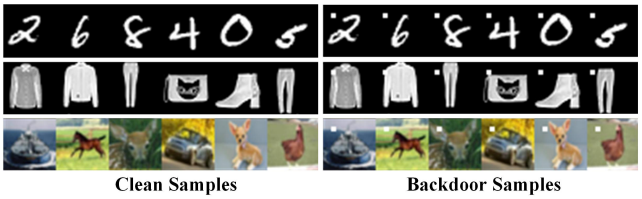


Figure 7: The clean samples and backdoor samples.

Our solution is to introduce $\mathcal{F}_m$ to optimize the training process for backdoor implantation. After the finetuning in the second stage, $\mathcal{F}_m$ has adapted to the encoding scheme of $\mathcal{F}_s$ and demonstrates good performance in both the main and backdoor attack tasks. Therefore, to some extent, $\mathcal{F}_m$ also contains information about $\mathcal{F}_s$ and can guide the training of $\mathcal{F}_c$, aligning with our training strategy in the first stage. Additionally, by constructing the loss function for the main task, we can enhance the consistency between $\mathcal{F}_c$ and $\mathcal{F}_m$ in terms of the main task.

Overall, in the Implant stage, our training process is as follows. Firstly, we regard $\mathcal{F}_c$ as a generator and introduce a discriminator $\mathcal{F}_d$. Thus, a generative adversarial network (GAN) system is formed among $\mathcal{F}_c$, $\mathcal{F}_d$, $Z_c$and Z_s. We freeze the parameters of $\mathcal{F}_s$ and train $\mathcal{F}_d$ and $\mathcal{F}_c$ using D_c and D_s. The loss functions are defined as follows:

$$L_d = log(1 - \mathcal{F}_d(Z_s)) + log(\mathcal{F}_d(Z_c)) \tag{6}$$

$$L_{dc} = log(1 - \mathcal{F}_d(Z_c)) \tag{7}$$

Next, we freeze the model parameters of $\mathcal{F}_m$ and train $\mathcal{F}_c$ using D_c. The loss function is defined as follow:

$$L^3_{mc} = L(\mathcal{F}_m(Z_c), Y_c) \tag{8}$$

Our training strategy aims to minimize the impact of the backdoor implantation process on the main task and ensure rapid convergence of the training results. Figure 6 presents the training results of the third stage.

Experiments

In this section, we will evaluate the effectiveness and generality of our proposed SFI backdoor attack framework.

Dataset	First Epochs	Second Epochs	Third Epochs	Shadow Epochs	Number Clients	Batch Size	D_c	D_s	Backdoor Sample	Target Label	Split Strategy	$\mathcal{F}_a$
MNIST	10	40	40	5	10	250	59500	500	50	1	Split B	with
Fashion	30	40	40	5	10	250	48750	1250	50	1	Split B	with
CIFAR10	50	40	50	5	10	250	47500	2500	100	1	Split B	with
CIFAR100	50	40	50	5	10	250	47500	2500	100	1	Split B	with

Table 1: The training parameters and configurations in our experiments.

Split	MNIST			Fashion			CIFAR10			CIFAR100		
	base	main	backcdoor	base	main	backdoor	base	main	backdoor	base	main	backdoor
Split A	0.993	0.958	0.885	0.932	0.899	0.955	0.957	0.916	0.868	0.584	0.528	0.895
Split B	0.991	0.943	0.928	0.926	0.875	0.974	0.939	0.882	0.918	0.561	0.514	0.912

Table 2: The attack performance for different dataset. The "base" represent the baseline of the main task (without attack).

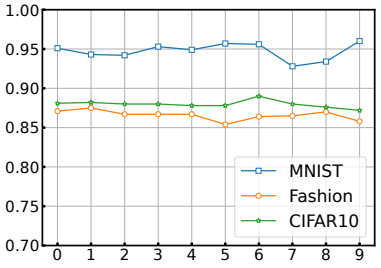


Figure 8: The impact of the target label on the main task performance. The horizontal axis represents the target labels set by the attacker.

Implementation Details

We conduct our experiments on four datasets: MNIST (Deng 2012), Fashion (Fashion-MNIST) (Xiao, Rasul, and Vollgraf 2017), CIFAR10, and CIFAR100 (Krizhevsky, Hinton et al. 2009). The shadow dataset on the server side is sampled from the original training data, while the remaining training data is evenly distributed among participating clients as their local training data. To construct the backdoor samples, we overlay a white square (backdoor trigger) on clean samples, as illustrated in Figure 7. We present the training hyperparameters and configurations in Table 1.

Our attack framework consists of five models: $\mathcal{F}_c$, $\mathcal{F}_s$, $\mathcal{F}_m$, $\mathcal{F}_a$, and $\mathcal{F}_d$. The main task of split learning involves training a ResNet18 model composed of $\mathcal{F}_c$ and $\mathcal{F}_m$. We employ two different split strategies. The split point of Split A is before the first ResBlock of ResNet18, and the split point of Split B is after the first ResBlock of ResNet18.

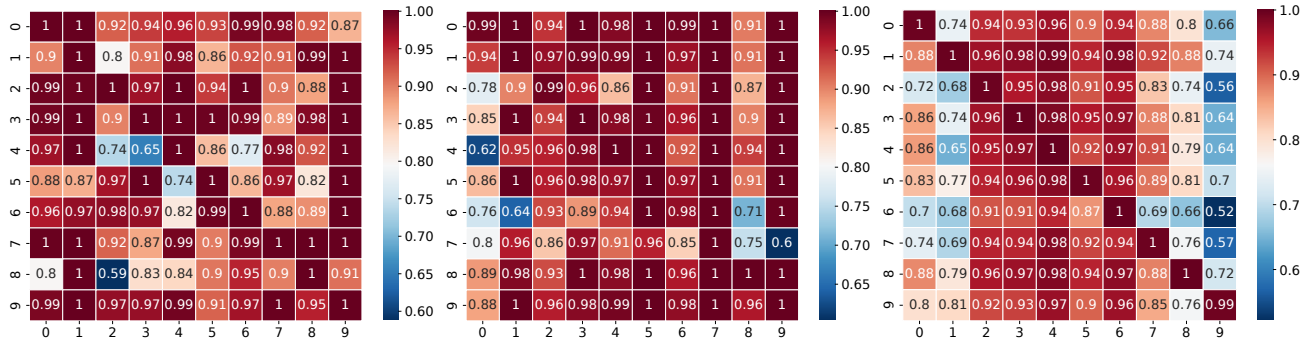
Results and Analysis

Target label. Considering the diverse attack requirements, the backdoor attack framework needs to support the customization of target labels. We recode the labels of the three datasets as 0 to 9, and the test results are shown in Figure 9. The horizontal axis represents the source labels of the backdoor samples, while the vertical axis represents the target labels set by the attackers. Our attack framework demonstrates high attack accuracy across all three datasets for various backdoor target labels. In addition, the model's performance on the main task suffers limited impact when changing the backdoor target label, as illustrated in Figure 8.

Split strategy. Split learning can adapt to different needs and scenarios by flexibly adjusting the split strategy. Different split strategies mean the adjustment of the network architecture so that it will have an impact on the training process and attack performance. We test two split strategies, Split A and Split B, and the experimental results are shown in Table 2. Overall, our proposed attack framework exhibits remarkable adaptability to different splitting strategies, delivering expected performance outcomes in both the main task and backdoor attack. Notably, slight variances can be observed between the two splitting strategies across different evaluation metrics. Due to the additional ResBlock in $\mathcal{F}_m$ of Split A compared to Split B, $\mathcal{F}_m$ (Split A) can encode more information, resulting in its advantage in the main task. Another observation is that $\mathcal{F}_c$ (Split B) has an additional ResBlock compared to Split A, providing sufficient encoding space to enhance sensitivity to backdoor samples. As a result, Split B demonstrates better performance in backdoor attacks.

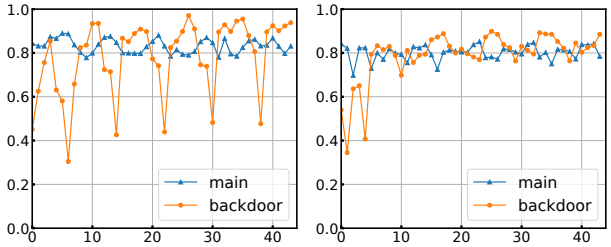
Data distribution. In split learning, data distribution is an essential factor affecting the experimental results. Figure 6 and Figure 10 respectively show the attack performance of the Implant stage under the iid and non-iid settings. We generate heterogeneous data across clients for non-iid settings by the Dirichlet-based data partition strategy. Generally, in the normal training process, the experimental outcomes and training efficiency in the non-iid setting are slightly inferior to the iid setting. Similarly, compared to the iid setting, the attack performance of SFI and the stability of the training process also decrease in the non-iid scenario. Overall, the main task is more sensitive to data distribution, and the backdoor attack still shows high accuracy on these two datasets.

Shadow dataset. Our attack framework assumes that the only background knowledge available to the attacker is the shadow dataset of limited data samples. This assumption significantly reduces the cost and barriers to launch the attack. To evaluate the impact of the number of shadow dataset



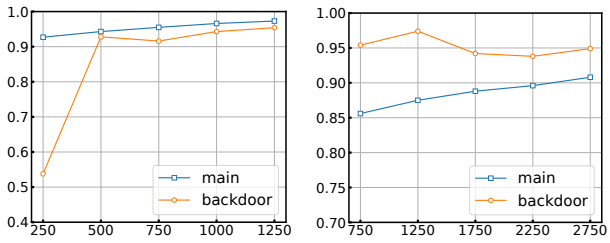
(a) MNIST dataset (b) Fashion dataset (c) CIFAR10 dataset

Figure 9: The performance of the backdoor attack ($\mathcal{F}_c$ and $\mathcal{F}_m$) for different target labels. The horizontal axis represents the source labels of the backdoor sample, while the vertical axis represents the target labels set by the attacker.



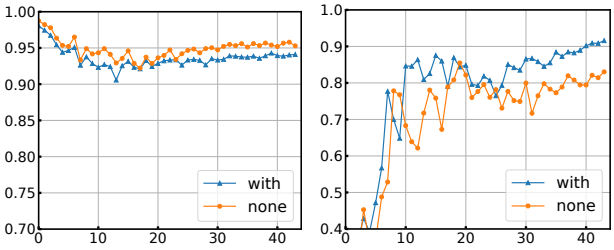
(a) MNIST dataset (b) Fashion dataset

Figure 10: The attack performance of non-iid settings (the Dirichlet-based data partition strategy with $\alpha = 0.3$).



(a) MNIST dataset (b) Fashion dataset

Figure 11: The impact of the number of shadow dataset samples on attack performance.



(a) The main task (b) Backdoor attack

Figure 12: The impact of auxiliary model on attack performance (MNIST dataset).

samples on attack performance, we test five sets of values on the two datasets, and the results are shown in Figure 11. We observe that the attacker only requires a minimal number of shadow dataset samples to achieve significant attack effectiveness. For the main task, the greater the number of shadow datasets, the more extensive the training $\mathcal{F}_m$ receives during the Finetune stage. Consequently, the impact on the main task during the Implant stage becomes less significant. It is worth noting that the number of backdoor samples used in our tests remains consistent. As the number of samples in the shadow dataset increases, the proportion of backdoor samples decreases, thus impacting the performance of backdoor attacks. The attack performance can be improved by appropriately increasing the number of backdoor samples.

Auxiliary model. Including the auxiliary model $\mathcal{F}_a$ in the initial two stages of the attack framework aims to enhance $\mathcal{F}_s$'s sensitivity to backdoor samples, thereby empowering $\mathcal{F}_c$ to gain more robust capabilities in encoding the backdoor samples during the Implant phase. Figure 12 illustrates the performance of the Implant stage with and without $\mathcal{F}_a$ concerning both the main task and backdoor attack. The experimental results show that the auxiliary model $\mathcal{F}_a$ significantly improves the performance of the backdoor attack, resulting in an enhancement of approximately 8% in attack effectiveness on the MNIST dataset. Furthermore, the impact of $\mathcal{F}_a$ on the model's performance in the main task is limited.

Summary and Future Work

In this paper, we first reveal the vulnerability of split learning to server-side backdoor attacks and propose a general backdoor attack framework named SFI. Our research demonstrates that even with limited background knowledge, a malicious server-side attacker can effectively leverage intermediate information during the training process to manipulate the optimization direction of client models. Various security risks in split learning expose vulnerabilities in its learning patterns and interaction protocol. Future, we try to build an efficient and secure split learning training mechanism.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (2023YFB3106900); in part by the National Natural Science Foundation of China (NSFC62372334).

References

Bagdasaryan, E.; and Shmatikov, V. 2020. Blind Backdoors in Deep Learning Models. In *USENIX Security Symposium*.

Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, 2938–2948. PMLR.

Creswell, A.; White, T.; Dumoulin, V.; Arulkumaran, K.; Sengupta, B.; and Bharath, A. A. 2018. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1): 53–65.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Erdogan, E.; Küpçü, A.; and Cicek, A. E. 2022. SplitGuard: Detecting and Mitigating Training-Hijacking Attacks in Split Learning. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society*, 125–137.

Erdoğan, E.; Küpçü, A.; and Çiçek, A. E. 2022. UnSplit: Data-Oblivious Model Inversion, Model Stealing, and Label Inference Attacks against Split Learning. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society*, 115–124.

Fu, C.; Zhang, X.; Ji, S.; Chen, J.; Wu, J.; Guo, S.; Zhou, J.; Liu, A. X.; and Wang, T. 2022. Label inference attacks against vertical federated learning. In *31th USENIX Security Symposium*, 1397–1414.

Fu, J.; Ma, X.; Zhu, B. B.; Hu, P.; Zhao, R.; Jia, Y.; Xu, P.; Jin, H.; and Zhang, D. 2023. Focusing on Pinocchio's Nose: A Gradients Scrutinizer to Thwart Split-Learning Hijacking Attacks Using Intrinsic Attributes. In *30th Annual Network and Distributed System Security Symposium*.

Gao, X.; and Zhang, L. 2023. PCAT: Functionality and Data Stealing from Split Learning by Pseudo-Client Attack. In *32th USENIX Security Symposium*.

Gao, Y.; Doan, B. G.; Zhang, Z.; Ma, S.; Zhang, J.; Fu, A.; Nepal, S.; and Kim, H. 2020a. Backdoor attacks and countermeasures on deep learning: A comprehensive review. *arXiv preprint arXiv:2007.10760*.

Gao, Y.; Kim, M.; Abuadbba, S.; Kim, Y.; Thapa, C.; Kim, K.; Camtepe, S. A.; Kim, H.; and Nepal, S. 2020b. End-to-end evaluation of federated learning and split learning for internet of things. *arXiv preprint arXiv:2003.13376*.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2): 1–210.

Kariyappa, S.; and Qureshi, M. K. 2022. ExPLoit: Extracting Private Labels in Split Learning. *arXiv preprint arXiv:2112.01299*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Li, O.; Sun, J.; Yang, X.; Gao, W.; Zhang, H.; Xie, J.; Smith, V.; and Wang, C. 2021. Label leakage and protection in two-party split learning. *arXiv preprint arXiv:2102.08504*.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3): 50–60.

Li, Y.; Jiang, Y.; Li, Z.; and Xia, S.-T. 2022. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

Liu, J.; and Lyu, X. 2022. Clustering Label Inference Attack against Practical Split Learning. *arXiv preprint arXiv:2203.05222*.

Nguyen, T. A.; and Tran, A. T. 2021. WaNet-Imperceptible Warping-based Backdoor Attack. In *International Conference on Learning Representations*.

Pasquini, D.; Ateniese, G.; and Bernaschi, M. 2021. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2113–2129.

Singh, A.; Vepakomma, P.; Gupta, O.; and Raskar, R. 2019. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145*.

Vepakomma, P.; Gupta, O.; Swedish, T.; and Raskar, R. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*.

Wenger, E.; Passananti, J.; Bhagoji, A. N.; Yao, Y.; Zheng, H.; and Zhao, B. Y. 2021. Backdoor attacks against deep learning systems in the physical world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6206–6215.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.

Zhang, Z.; Panda, A.; Song, L.; Yang, Y.; Mahoney, M.; Mittal, P.; Kannan, R.; and Gonzalez, J. 2022. Neurotoxin: Durable backdoors in federated learning. In *International Conference on Machine Learning*, 26429–26446. PMLR.

Zhang, Z.; Pinto, A.; Turina, V.; Esposito, F.; and Matta, I. 2023. Privacy and Efficiency of Communications in Federated Split Learning. *arXiv preprint arXiv:2301.01824*.