FedDefender: Backdoor Attack Defense in Federated Learning

Waris Gill
Virginia Tech
Blacksburg, USA
waris@vt.edu

Ali Anwar
University of Minnesota
Minneapolis, USA
aanwar@umn.edu

Muhammad Ali Gulzar
Virginia Tech
Blacksburg, USA
gulzar@cs.vt.edu

ABSTRACT

Federated Learning (FL) is a privacy-preserving distributed machine learning technique that enables individual clients (*e.g.,* user participants, edge devices, or organizations) to train a model on their local data in a secure environment and then share the trained model with an aggregator to build a global model collaboratively. In this work, we propose FedDefender, a defense mechanism against targeted poisoning attacks in FL by leveraging differential testing. FedDefender first applies differential testing on clients' models using a synthetic input. Instead of comparing the output (predicted label), which is unavailable for synthetic input, FedDefender fingerprints the neuron activations of clients' models to identify a potentially malicious client containing a backdoor. We evaluate FedDefender using MNIST and FashionMNIST datasets with 20 and 30 clients, and our results demonstrate that FedDefender effectively mitigates such attacks, reducing the attack success rate (ASR) to 10% without deteriorating the global model performance.

CCS CONCEPTS

• **Software testing and debugging**; • **Software safety**;

KEYWORDS

federated learning, testing, backdoor attack, poisoning attack, differential testing, deep learning, fault localization

ACM Reference Format:
Waris Gill, Ali Anwar, and Muhammad Ali Gulzar. 2023. FedDefender: Backdoor Attack Defense in Federated Learning. In *Proceedings of the 1st International Workshop on Dependability and Trustworthiness of Safety-Critical Systems with Machine Learned Components (SE4SafeML '23), December 4, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3617574.3617858

1 INTRODUCTION

Federated Learning (FL) trains a global model on decentralized data from multiple clients without directly accessing their individual data samples. FL improves model accuracy by leveraging the combined data from multiple clients and also improves privacy by keeping individual data samples on the clients' devices. However, the decentralized nature of FL makes it vulnerable to targeted poisoning attacks (often called backdoor attacks). In such attacks, an adversarial client manipulates its training data to inject a backdoor into a global model. Since the server does not have access to the raw training data of the clients, such attacks remain hidden until a trigger is injected into the input. Therefore, it is highly challenging to detect and defend against backdoor attacks in FL [8, 12].

Prior work [10] on defending against targeted poisoning attacks in FL has focused on using norm clipping (*NormClipping*) to detect and mitigate these attacks. Norm clipping involves computing the norms of model updates received from clients and rejecting updates that exceed a certain threshold. This technique has been shown to be effective in some cases, but it has limitations. For example, if an attacker carefully crafts the attack such that the norm of the gradient is not noticeably large, the norm clipping approach will not be effective in detecting the attack. Therefore, alternative approaches are needed to defend against targeted poisoning attacks.

Contribution and Key Insight. In this work, we propose FedDefender, a defense against backdoor attacks in federated learning by leveraging differential testing for FL [4]. FedDefender minimizes the impact of a malicious client on the global model by limiting its contribution to the aggregated global model. Instead of comparing the predicted label of an input, which is often unavailable in FL, FedDefender fingerprints the neuron activations of clients' models on the same input and uses differential testing to identify potential malicious clients. Our insight is that since clients in FL have homogeneous models trained on similar concepts, their neuron activations should have some similarities on a given input [4]. At the central server, if a client's model displays neuron activation patterns that significantly differ from other clients (*i.e.,* majority of clients), such a client's model may contain a trigger pattern and can be flagged as potentially malicious.

Evaluations. We evaluate FedDefender with 20 and 30 FL clients on MNIST and FashionMNIST datasets. Our results demonstrate that compared to the norm clipping defense [10], FedDefender effectively defends against backdoor attacks and reduces the attack success rate (ASR) to 10% without negatively impacting the global model accuracy. FedDefender's artifact is available at https://github.com/warisgill/FedDefender.

2 BACKGROUND AND RELATED WORK

Federated Learning. In Federated Learning (FL), multiple *clients* (*e.g.,* mobile devices, smart home devices, and autonomous vehicles) locally train models on their private training data. The trained client's model is sent back to a *central server* (also called an *aggregator*). A client's model comprises a collection of *weights* connecting *neurons* in a neural network. All client models work on structurally (same number of neurons and layer) same neural network. After the participating clients' models are received, the aggregator uses a *fusion* algorithm to merge all models into a single *global model.* A *round* in FL starts with the client's training and ends once a global model is constructed. Federated Averaging (FedAvg) [7] is a popular

SE4SafeML '23, December 4, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0379-9/23/12.
https://doi.org/10.1145/3617574.3617858

fusion algorithm that uses the following equation to build a global model using the client's models at each round.

$$W_{global}^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} W_k^{(t)} \quad (1)$$

$W_k^{(t)}$ and n_k represent weights and size of training data of client k in a given round t, respectively. The variable n represents the total number of data points from all clients, and it is calculated as $n = \sum_{k=1}^{K} n_k$. At the end of the round, the global model is sent back to all participating clients to be used as a *pretrained* model in their local training during the next round. A *malicious client* sends its incorrect model after injecting a backdoor during its local training to manipulate the global model.

Differential Testing. Differential testing is a software testing technique. It executes two or more comparable programs on the same test input and compares resulting outputs to identify unexpected behavior [5]. In prior work, it is used to find bugs in compilers [14], deep neural networks [9], and faulty clients in FL[4].

Backdoor Attack and Defense. Backdoor attacks in the context of computer vision refer to a specific type of malicious behavior in which an attacker injects a "backdoor" into a machine learning (ML) model during its training [11]. This backdoor allows the attacker to gain control over the model by providing a specific input that triggers the model to behave in a way that is beneficial to the attacker. This type of attack is particularly concerning as models are often used for tasks such as object recognition, and the ability to manipulate these models can have significant real-world consequences. For example, an attacker could train a model to recognize a stop sign but also include a hidden trigger that causes misclassification, leading to unsafe situations in the real world.

In FL, a malicious client k can inject a backdoor to the global model (W^{t+1}) by manipulating its local model $W_k^{(t)}$. Prior approaches [8, 12] propose defenses by changing the underlying FL training protocol (*e.g.,* changes in the FedAvg protocol). Such defenses require special alterations to work with other FL training protocols such as FedProx [6] and FedAvg [7]. Sun et al. [10] propose norm clipping to detect and mitigate these attacks. Norm clipping can degrade the performance of a global model, and it can be easily bypassed with carefully crafted attacks. Therefore, alternative approaches are needed that can be integrated with any fusing algorithm (*e.g.,* FedAvg [7], FedProx [6]) without requiring any changes to fusion protocols and, at the same time, do not impact the performance of the global model while still protecting against backdoor attacks.

3 THREAT MODEL

We consider a single malicious client (*i.e.,* attacker) participating in each round (t). The malicious client k injects a square trigger pattern (4 x 4) to its n_k training images to manipulate its local model ($W_k^{(t)}$) during local training. The attacker can increase the strength of a backdoor attack X times by scaling up its number of training data points n_k (*e.g.,* $n_k \leftarrow n_k \cdot 20$) to successfully inject the backdoor into the global model (W^{t+1}) during aggregation (Equation 1). The goal of the attacker in this threat model is to gain control over the federated learning model by injecting a backdoor trigger and using it to manipulate the model's behavior.

4 FEDDEFENDER DESIGN

Algorithm 1: FedDefender Defense

```
Input: Let $client2weights = \{w_1, w_2, ... w_k\}$ be a dictionary that maps k clients to
       their models' parameters
Input: Let $N = \{n_1, n_2, ... n_k\}$ be number of training examples of k clients
Input: Let $test\_inputs$ be a list of inputs
Input: Let $\theta$ be a threshold for malicious confidence
Output: $W^{t+1}$: global model for the next round
$client2mal\_confidence = Dictionary()$ // An empty dictionary
$min\_n_k = min(N)$ // Get a minimum number of training examples
    among k clients
for each $input\_i \in test\_inputs$ do
    // Find the potential malicious client using FL
       differential testing technique [4]
    $potential\_mal\_client =$
      $FL\_DifferentialTesting(client2weights, input\_i, 0)$
    // Increment the confidence of a potential malicious client
    $client2mal\_confidence[potential\_mal\_client] += 1$
// Defense by restricting potential malicious clients
    contribution
for each $client \in client2mal\_confidence$ do
    // Normalize the confidence of each client
    $client2mal\_confidence[client] =$
      $client2mal\_confidence[client]/len(test\_inputs)$
    // If threshold satisfied, discard malicious client's
       contribution by setting their number of training
       examples to 0
    if $client2mal\_confidence[client] > \theta$ then
        $N[client] = 0$
    else
        $N[client] = int(min\_n_k * (1 - client2mal\_confidence[client]))$
            // Reduce training examples if the client seems
            less malicious to penalize their contribution to
            the global model.
$W^{t+1} = FedAvg(client2weights, N)$ // Compute the global model using
    FedAvg
return $W^{t+1}$
```

To achieve optimal performance of the global model and protect its integrity, it is critical to correctly identify the potential malicious clients and restrict their participation in the global model W^{t+1} before the aggregation step (Equation 1). Access to clients' data is prohibited in FL, and collecting new test data at the central server has its own challenges. Such challenges make existing backdoor detection techniques [11] impractical. Thus, backdoor detection in FL requires a novel solution to mitigate the backdoor attack without any dependence on real-world test data.

Differential Testing FL Clients. Gill et al. [4] propose a differential technique to find faulty clients in an FL round training without requiring access to real-world test inputs. It generates inputs randomly at the central server and compares the behaviors of clients' models at the neuron level to localize a faulty client. The internal neuron values of the models are used as a fingerprint of the behavior on the given input, and a client is flagged as malicious if its behavior deviates significantly from the majority of the clients. The key insight is that the behavior of a malicious client's model will be different from that of benign clients, as malicious executions are inherently different from correct ones. We use a neuron activation threshold equal to zero to profile the behavior (*i.e.,* neuron activations) of a client model.

FedDefender adapts differential testing technique for FL [4] to detect behavioral discrepancies among clients' models, with the aim of identifying potential malicious clients in a given FL training

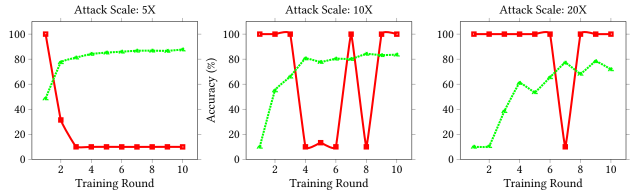
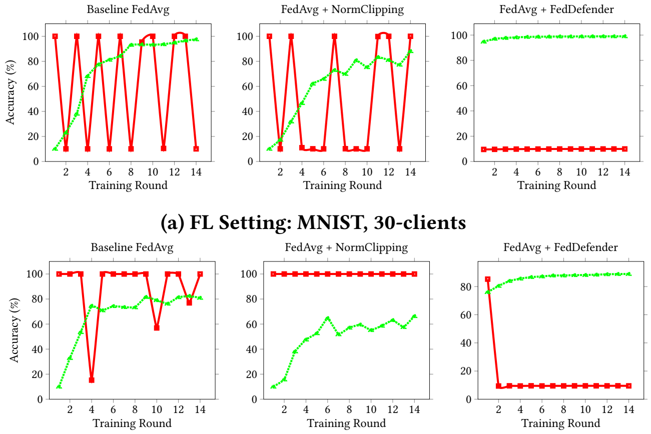


Figure 1: The scaling factor increases the strength of the malicious client by increasing the number of training examples, n_k, by a factor of X. This enhances the chances of successfully injecting a backdoor in the global model W^{t+1}.



(a) FL Setting: MNIST, 30-clients

(b) FL Setting: FashionMNIST, 30-clients

Figure 2: Comparison of FedDefender, with the baseline FedAvg and NormClipping defense mechanisms. Figures indicate that FedDefender successfully mitigates the attack and lowers the ASR close to 10% .

round. Algorithm 1 outlines the defense strategy of FedDefender against backdoor attacks in FL. The inputs to Algorithm 1 include the clients' models (*client2weights*), a list containing the number of training examples for each client (N), a set of randomly generated test inputs (*test_inputs*), and a threshold for malicious confidence θ. FedDefender first employs the differential execution technique, as outlined in [4], to identify a potential malicious client on each input. It then updates the corresponding client's malicious score (lines 3-5). Subsequently, FedDefender limits the contribution of a client if its malicious confidence exceeds the specified threshold θ (lines 6-11). Finally, the global model is computed using the updated contribution of clients (line 12). As an illustration, consider a scenario in which ten clients are participating in a given FL training round, the malicious threshold is set at 0.5, and 100 test inputs are generated. FedDefender computes the malicious confidence of all clients. Clients 1, 3, and 7 have malicious confidence scores of 20/100, 60/100, and 20/100, respectively. The remaining clients have a malicious confidence score of zero. FedDefender discards the contribution of client 3 as it exceeds the malicious threshold and accordingly limits the contributions of the other clients.

5 EVALUATION

We evaluate FedDefender on (1) Attack Success Rate (ASR) [11] and (2) classification performance of the global model.

Dataset, Model, FL Framework. We use MNIST [3] and FashionMNIST [13] datasets. Each dataset contains 60K training and 10K testing grayscale, 28x28 images spanning ten different classes. The data is randomly distributed without any overlapping data points among FL clients. Each client trains a convolutional neural network (CNN). The CNN architecture is outlined in [1]. We set the learning rate to 0.001, epochs equal to 5 and 15, batch size of 32, and trained each configuration for at least 10 rounds. We implement our approach in Flower FL framework [2]. We run our experiments on AMD 16-core processor with 128 GB RAM.

Evaluation Metrics. We used the attack success rate (ASR) [11] and classification accuracy of the global model to compare FedDefender with norm clipping defense [10].

Backdoor Attack Strength. The strength of a backdoor attack (on the global model W^{t+1}) can be evaluated by considering the injection of a 4x4 trigger pattern into the training data of a malicious client, as well as the scaling of the number of examples used for such injection. Figure 1 demonstrates the effect of varying attack scales on the attack success rate (ASR) in an FL configuration consisting of 20 clients with the FashionMNIST dataset. Without scaling, *i.e., Attack Scale* = 5×, a malicious client is unable to inject a backdoor into the global model successfully. For the remaining experiments, a 20× scale is used to represent the maximum strength of the backdoor attack.

Backdoor Defense Evaluation. We compare FedDefender with the baseline Federated Averaging (FedAvg) algorithm (*i.e.,* without any defense) [7] and the *NormClipping* defense mechanism [10], using 30 FL clients configurations. The MNIST and FashionMNIST datasets are used in these experiments. Each setting is trained for 14 rounds, with 5 epochs in each round. The results of these experiments are illustrated in Figure 2, with the x-axis representing the number of training rounds and the y-axis representing the accuracy. The attack success rate (ASR) and classification accuracy are used to compare FedDefender with the baseline and *NormClipping* defense mechanisms. A lower ASR indicates that the malicious client is unable to manipulate the global model behavior using its backdoor. As shown in Figures 2, the *NormClipping* defense fails to provide any defense against the backdoor attack and also negatively impacts the global model's (W^{t+1}) classification accuracy. In contrast, FedDefender successfully mitigates the attack and lowers the ASR close to 10% without deteriorating the global model's classification accuracy.

Malicious Confidence Threshold (θ). The impact of the malicious confidence threshold (θ) in Algorithm 1 on the mitigation of the backdoor attack is also examined. Figure 3 shows the results of this analysis, using an FL configuration of 20 clients trained on the MNIST dataset. Each client model is trained for 15 epochs. Figure 3 illustrates that unless the potential malicious client is penalized aggressively, FedDefender is incapable of mitigating the attack. To aggressively penalize a client, the client's contribution is ignored before aggregation (lines 8-11 of Algorithm 1).

Takeaway: FedDefender successfully protects against backdoor attacks without impacting the global model accuracy.

FedDefender False Positive Rate. We evaluate the false positive rate to assess the impact of FedDefender on the global model's classification accuracy using a federated learning (FL) setting of 20 clients and the FashionMNIST dataset. In this scenario, all clients are benign, that is, there is no malicious client present. As shown in Figure 4, FedDefender hardly produces any false positives and demonstrates similar performance as the baseline Federated Averaging (FedAvg) and *NormClipping* defense mechanisms.

> **Takeaway:** FedDefender does not impact the global model accuracy, even if there is no malicious client.

Threat to Validity. To address potential threats to external validity, we perform experiments on two standardized FL datasets. Additionally, to mitigate potential threats arising from randomness in the FedDefender's random input generation, we evaluate each configuration on at least 100 random test inputs to compute the malicious confidence of a client. Despite these measures, certain threats to the validity of the experiments, such as variations in data distribution across clients, neuron activation threshold (default is zero), size of random test input, and type of convolutional neural networks (CNNs) may still exist. Future research will explore these potential threats in greater detail.

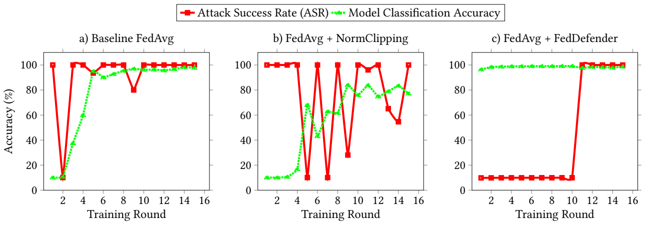


Figure 3: Evaluation of the impact of the malicious confidence threshold. FedDefender is unable to mitigate the attack if the potential malicious client is not aggressively penalized.

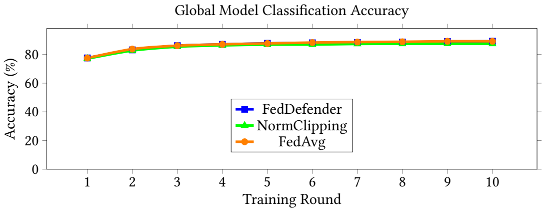


Figure 4: FedDefender performs similarly to FedAvg and NormClipping without penalizing benign clients.

6 FUTURE WORK AND CONCLUSION

Future Work. In future work, we propose to evaluate the potential of FedDefender by assessing its performance under various FL training settings. This could include varying the number of malicious clients, the number of training epochs, and data distribution across clients (*i.e.,* non-IID data distributions). Additionally, efforts could be made to further improve the detection capabilities of FedDefender, allowing precise identification of *multiple* malicious clients and reverse engineering their corresponding backdoor trigger patterns.

Another avenue of research would be to analyze the aggregation overhead of FedDefender compared to traditional aggregation protocols in FL. Extending the applicability of FedDefender to other model architectures, such as Transformers, which are commonly used in natural language processing tasks and speech recognition models, could be explored. Finally, incorporating realistic synthetic test inputs generated using generative adversarial networks (GANs) into the evaluation process could provide further insight into the performance of FedDefender.

Conclusion. Our position is that traditional software testing principles have matured over the years and have provably improved the state of testing software; therefore, FL should benefit from similar advancements. In this work, we propose FedDefender, a defense mechanism against targeted poisoning attacks in FL that utilizes random test generation with differential testing. We demonstrate that FedDefender effectively detects and mitigates such attacks, reducing the ASR to 10% without negatively impacting the global model accuracy. Our results show that FedDefender is more effective than the norm clipping defense and the baseline Federated Averaging (FedAvg) algorithm.

REFERENCES

[1] 2023. Training a Classifier — PyTorch Tutorials 1.13.1+cu117 documentation. https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html. (Accessed on 01/26/2023).

[2] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D Lane. 2020. Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390* (2020).

[3] Dan C Cireşan, Ueli Meier, Jonathan Masci, Luca M Gambardella, and Jürgen Schmidhuber. 2011. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183* (2011).

[4] Waris Gill, Ali Anwar, and Muhammad Ali Gulzar. 2023. FedDebug: Systematic Debugging for Federated Learning Applications. In *Proceedings of the 45th International Conference on Software Engineering* (Melbourne, Victoria, Australia) *(ICSE '23)*. IEEE Press, 512–523. https://doi.org/10.1109/ICSE48619.2023.00053

[5] Muhammad Ali Gulzar, Yongkang Zhu, and Xiaofeng Han. 2019. Perception and practices of differential testing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 71–80.

[6] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* 2 (2020), 429–450.

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

[8] Mustafa Safa Ozdayi, Murat Kantarcioglu, and Yulia R Gel. 2021. Defending against backdoors in federated learning with robust learning rate. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9268–9276.

[9] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*. 1–18.

[10] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. 2019. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963* (2019).

[11] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 707–723.

[12] Shuaiqi Wang, Jonathan Hayase, Giulia Fanti, and Sewoong Oh. 2022. Towards a Defense against Backdoor Attacks in Continual Federated Learning. *arXiv preprint arXiv:2205.11736* (2022).

[13] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.* arXiv:cs.LG/1708.07747 [cs.LG]

[14] Xuejun Yang, Yang Chen, Eric Eide, and John Regehr. 2011. Finding and understanding bugs in C compilers. In *Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation*. 283–294.

Received 2023-07-04; accepted 2023-08-10