TimberStrike: Dataset Reconstruction Attack Revealing Privacy Leakage in Federated Tree-Based Systems

Marco Di Gennaro*
Politecnico di Milano
Milan, Italy
marco.digennaro@polimi.it

Giovanni De Lucia*
Politecnico di Milano
Milan, Italy
giovanni.delucia@mail.polimi.it

Stefano Longari
Politecnico di Milano
Milan, Italy
stefano.longari@polimi.it

Stefano Zanero
Politecnico di Milano
Milan, Italy
stefano.zanero@polimi.it

Michele Carminati
Politecnico di Milano
Milan, Italy
michele.carminati@polimi.it

Abstract

Federated Learning has emerged as a privacy-oriented alternative to centralized Machine Learning, enabling collaborative model training without direct data sharing. While extensively studied for neural networks, the security and privacy implications of tree-based models remain underexplored. This work introduces TimberStrike, an optimization-based dataset reconstruction attack targeting horizontally federated tree-based models. Our attack, carried out by a single client, exploits the discrete nature of decision trees by using split values and decision paths to infer sensitive training data from other clients. We evaluate TimberStrike on State-of-the-Art federated gradient boosting implementations across multiple frameworks, including Flower, NVFlare, and FedTree, demonstrating their vulnerability to privacy breaches. On a publicly available stroke prediction dataset, TimberStrike consistently reconstructs between 73.05% and 95.63% of the target dataset across all implementations. We further analyze Differential Privacy, showing that while it partially mitigates the attack, it also significantly degrades model performance. Our findings highlight the need for privacy-preserving mechanisms specifically designed for tree-based Federated Learning systems, and we provide preliminary insights into their design.

Keywords

Federated Learning, Privacy Attacks, Dataset Reconstruction Attack, Gradient Boosting Decision Trees

1 Introduction

Federated Learning (FL) [41] is a Machine Learning (ML) paradigm in which decentralized nodes can collaboratively train a model. Specifically, these nodes train a shared *global model* under the coordination of a central server, known as *Parameter Server (PS)*, while keeping their local data private. There are different types of Federated Learning. Between them, we focus on *horizontal* FL [29], where clients (or nodes) have different datasets but share the same feature space. FL is considered an alternative to traditional centralized ML training in privacy-sensitive domains, such as healthcare [23, 43]. The adoption of FL in applications like healthcare [49] and finance [1] is largely driven by data-sharing regulations, such as the European Union's General Data Protection Regulation (GDPR) [55] and the United States' Health Insurance Portability and Accountability Act (HIPAA) [7]. However, even if the paradigm does not involve data exposure, FL does not always guarantee the privacy of training data. Indeed, several studies have demonstrated that attackers can infer sensitive information from the exchanged model updates [26, 68], such as individual training samples or specific dataset properties.

Originally designed for Artificial Neural Networks (ANNs), FL has been extended to tree-based models [33, 39, 40, 53], given their strong performance on tabular data [35]. In particular, works proposing federated tree-based systems adapt FL settings to ensembles of Decision Trees (DTs), such as Gradient Boosting Decision Trees (GBDT) [20] and eXtreme Gradient Boosting (XGBoost) [10].

To the best of our knowledge, unlike ANNs, for which several privacy attacks and defenses have been proposed for both tabular [54] and non-tabular data [68], the privacy of tree-based models in FL remains underexplored. Prior works [33, 48] have addressed privacy concerns in federated tree-based systems by adapting existing privacy defenses such as Differential Privacy (DP) [17, 27], Secure Multi-Party Computation (MPC) [34], and Homomorphic Encryption (HE) [2]. Additionally, some studies have explored privacy attacks against tree-based *vertical* FL systems [12, 38, 52]. In contrast, the State-of-the-Art (SotA) lacks an in-depth examination of privacy attacks in tree-based horizontal FL scenarios. This gap in the SotA is relevant given the strong appeal of tree-based models for their interpretability and high performance on tabular datasets, which are especially prevalent in healthcare and other critical sectors where the horizontal FL paradigm is widely used.

In this work, we address this gap by proposing TimberStrike, a novel optimization-based dataset reconstruction attack that exploits privacy vulnerabilities in horizontally federated tree-based models. Our primary objective is to demonstrate the vulnerability to reconstruction attacks of the most promising variants of tree-based FL systems, implemented by well-known frameworks such as Flower [4] and NVFlare [48]. Regardless of the approach, we aim to demonstrate that the attributes that make tree-based models attractive for FL systems, such as their discrete split values and

*These authors contributed equally to this research.

Proceedings on Privacy Enhancing Technologies 2025(4), 566–584

https://doi.org/10.56553/popets-2025-0145

explicit decision paths, effectively expose them to privacy leakage in collaborative scenarios. Our attack is formalized and evaluated on federated gradient boosting models (e.g., GBDT and XGBoost), as these are supported by several widely adopted frameworks. We consider a threat model in which the adversary acts as an *honest-but-curious client*, seeking to steal other clients' training data by reconstructing it, without disrupting the training process to remain undetected. Our intuition is that an adversary with access to the trees built by other clients may use the splitting criteria generated from their training datasets to infer them. The attack consists of two main phases. The first one, namely *First-Tree Probing*, targets the first tree of the victim client and infers fundamental information, such as the number of samples in the training set and the label distribution. By extracting this information and following the tree splits, the adversary can generate an initial version of the reconstructed dataset. The second phase, namely *Feature Range Inference*, refines the feature ranges in the reconstructed dataset, improving the reconstruction of the victim's data by solving an optimization problem for each subsequent victim's tree. We adapt our attack methodology to four different federated XGBoost variants and one GBDT implementation. We consider three approaches implemented in the Flower framework [4] (Bagging, Cyclic, and FedXGBllr [39]), the histogram-based implementation in NVFlare [48], and a standalone framework called FedTree [33]. Adapting to the diverse aggregation mechanisms and implementation strategies of these systems constitutes a central challenge of our work. Finally, after analyzing SotA privacy defenses, we offer insights into how horizontal tree-based FL systems should be designed to be resilient against reconstruction attacks.

We experimentally evaluate our approach on two healthcare datasets: Stroke Prediction [24] and Pima Indians Diabetes [14]. We demonstrate that the State-of-the-Art (SotA) in horizontal tree-based FL, implemented in the most popular framework, is vulnerable to the TimberStrike attack. Specifically, we show that TimberStrike can reconstruct a significant portion of the target dataset, achieving a Reconstruction Accuracy (RA) consistently above 73.05% on the Stroke dataset across all features and implementations, and up to 95.63% when considering only the most important features. We further analyze the impact of each attack phase and its dependency on the algorithms' hyperparameters. Additionally, we show that while classical privacy defenses like DP reduce attack effectiveness, they fail to fully mitigate it and significantly degrade model performance, making the privacy-utility trade-off difficult to manage. Finally, we provide preliminary insights into the design of future privacy-preserving horizontal tree-based FL systems by examining the information leveraged by our attack.

Open Source and Ethics. We release the attack code for all evaluated systems[1]. Ethical considerations are discussed in Section A.

Our main contributions to the SotA are the following:

- We propose an optimization-based dataset reconstruction attack targeting tree-based horizontal FL systems. To the best of our knowledge, this is the first work to investigate reconstruction attacks in this specific setting. Our attack allows an honest-but-curious client-side adversary to infer other clients' training data by leveraging the exchanged model updates. Importantly, the proposed method is compatible with several SotA tree-based FL frameworks.
- We demonstrate that existing defense mechanisms—when compatible with our threat model—either fail to mitigate the attack or incur a substantial loss in model utility. This highlights the need for robust privacy-preserving mechanisms in tree-based FL systems.
- We provide some preliminary insights into the design principles of an ideal tree-based FL system that is robust against such reconstruction attacks.

[1] https://github.com/necst/TimberStrike

2 Background

In this section, we introduce the main concepts needed to understand our work. Indeed, we discuss the primers on Gradient Boosting Decision Trees (GBDT), eXtreme Gradient Boosting (XGBoost), Federated Learning, and federated tree-based systems. For the latter, we describe the SotA implementations for which we design a dataset reconstruction attack in this work. In addition, we provide background on Differential Privacy in Section B.

2.1 Primer on GBDT and XGBoost

To understand our dataset reconstruction attack, we first establish the key concepts of XGBoost and gradient boosting.

Decision Trees (DTs). A DT [47] is a tree-based model used for *classification* and *regression* tasks. It consists of *nodes* representing decision rules, *branches* representing possible outcomes, and *leaves* containing final predictions. The construction of a DT involves recursively splitting the training dataset based on feature values to maximize information gain or minimize impurity measures such as the Gini index or entropy. The final output of a DT is determined by the values in its leaf nodes (*leaf values*).

2.1.1 Gradient Boosting Decision Trees and XGBoost. GBDT [20] is an ensemble learning method that trains DTs sequentially, with each new tree correcting the residual errors of the previous ones. After training all trees, the *learning rate* determines each tree's contribution. XGBoost [10] is an optimized implementation of GBDT that improves training efficiency by incorporating L_1 and L_2 regularization, as well as parallelizing tree construction and tree pruning. Below, we introduce key definitions related to XGBoost and GBDT.

Base Score and Initial Predictions. The *base score* is a global model parameter. It is the global bias of the model, i.e., the initial prediction before any trees are trained. This value is crucial in TimberStrike since it influences Hessian calculations in the first trained tree, which we exploit for dataset reconstruction.

Gradient and Hessian Computation. Each data point contributes to training a new tree through the gradient and Hessian values that are then used to compute the gain for a certain split:

$$g_i = \partial_{\hat{y}_i{}^{(t-1)}} l(y_i, \hat{y}_i{}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}_i{}^{(t-1)}} l(y_i, \hat{y}_i{}^{(t-1)}), \tag{1}$$

where l is the loss function, y_i is the true label, $\hat{y}_i{}^{(t-1)}$ is the prediction of the i-th sample at iteration $t-1$, g_i (gradient from sample i) measures how much a sample's prediction needs to be corrected and h_i (Hessian value of sample i) determines confidence in the correction. Finally, the total Hessian value and gradient at a node are

respectively $H = \sum_{i=1}^{N} h_i$ and $G = \sum_{i=1}^{N} g_i$ where N is the number of samples assigned to that node.

Histogram-Based Hessian computation. It is possible to use histogram-based optimization for split identification. For each feature, continuous feature values are discretized into histogram bins, reducing memory usage and computational cost. A small number of split points is proposed. The algorithm accumulates the gradient and Hessian values within each bin, enabling rapid computation of split gains and improving the efficiency of tree construction.

Tree Construction and Leaf Assignments. XGBoost trees are built using a histogram-based approach that selects the best split by maximizing the gain:

$$\text{Gain} = \frac{1}{2}\left(\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right) - \gamma, \tag{2}$$

where G_L, H_L, G_R, and H_R are total gradients and Hessians for the left and right child nodes, while λ and γ are regularization parameters. At inference time, each sample follows a decision path dictated by its feature values, eventually landing in a leaf node.

2.2 Federated Learning

Federated Learning (FL) [41] is a learning paradigm in which multiple parties collaboratively train an ML model while keeping data decentralized, ensuring it remains on client devices. FL typically adopts a *client-server* architecture, relying on a central server, commonly referred to as the Parameter Server (PS). Clients train a model on their local datasets and send the resulting updates to the server. The server then aggregates these updates and returns an updated global model, which the clients use for subsequent training rounds or predictions. Fully decentralized architectures [6], where clients exchange and aggregate model updates with peers, have also emerged, but are beyond the scope of this work.

Federated Learning Classification. FL can be classified into three categories [29, 36]: *horizontal*, *vertical*, and federated *transfer learning*. In *horizontal* FL, clients share the same feature space but have different row samples. In *vertical* FL, clients have the same row samples but different feature spaces. Federated *transfer learning*, on the other hand, occurs when clients differ in both feature spaces and row samples. In this work, we target horizontal FL systems.

2.3 Federated Tree-Based Systems

Recent studies demonstrate that FL can be applied to tree-based models [8, 56, 61, 62]. For instance, researchers proposed FL systems based on models like Gradient Boosting Decision Trees (GBDT) [33], XGBoost [64], and Random Forest (RF) [16, 25]. In this work, we focus on several federated gradient boosting implementations integrated into widely adopted frameworks, including Flower Bagging and Cyclic [4], NVFlare's histogram aggregation [48], FedXGBllr [39], and FedTree [33]. Our focus is driven by the unique vulnerabilities of GBDT in the federated setting. Indeed, GBDT learns trees sequentially by leveraging detailed gradient statistics to refine prior errors, creating a richer and more exploitable attack surface for data reconstruction. Since our attack targets these specific mechanisms, RFs and simpler DTs are outside our work scope.

Flower XGBoost Bagging. The server distributes the global model to each client in the environment. The clients then use this global model as a starting point for the next boosting round. In particular, each client trains a new local tree based on the prediction obtained from the global model, then it sends this tree to the server, which aggregates them by concatenation in the order of arrival. This process repeats until the desired number of trees (or training rounds) is achieved. Such an approach is implemented in the Flower framework. Finally, the schema in Figure 1 summarizes the algorithm.

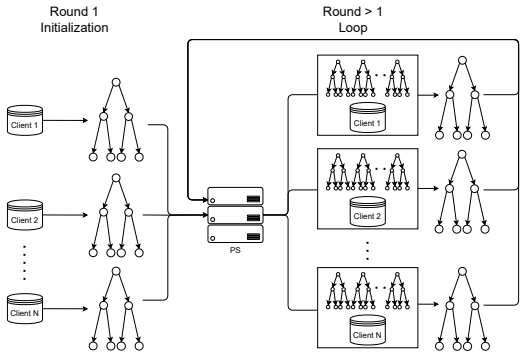


Figure 1: Flower Bagging algorithm schema.

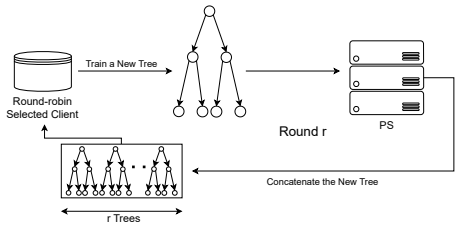


Figure 2: Flower Cyclic algorithm schema.

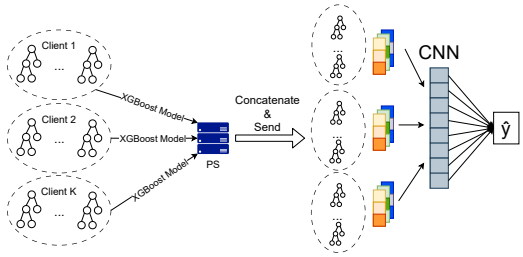


Figure 3: FedXGBllr algorithm schema.

Flower XGBoost Cyclic. In this protocol, depicted in Figure 2, the global tree ensemble is sequentially updated by different clients. At each round, the server sends the current ensemble to a selected client in a round-robin manner. The client then trains and concatenates a new tree (or trees for multiclass classification) to the ensemble before returning it to the server. This iterative process continues until the ensemble reaches a predefined number of trees.

FedXGBllr [39]. As shown in Figure 3, this protocol introduces a two-phase training strategy to federate XGBoost. In the first phase (first round), clients independently train XGBoost models and send them to the server, which aggregates (concatenation) them into a

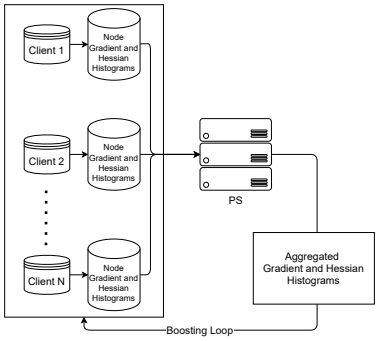


Figure 4: NVFlare Histogram-based algorithm schema.

global ensemble. The second phase (all subsequent rounds) involves constructing a dataset where each sample is represented by the leaf values it reaches across all trees in the aggregated ensemble. This dataset is then used as input to a federated one-dimensional Convolutional Neural Network (CNN 1-D), which is trained to predict the sample labels. The intuition is that the learned parameters of the CNN correspond to the learning rates assigned to each tree.

NVFlare - XGBoost Histogram Aggregation. Developed by NVIDIA, NVFLARE supports federated XGBoost training [48] by directly federating the XGBoost library. Among the implemented variants, NVFLARE enables histogram aggregation of gradient and Hessian statistics. Unlike the previously described mechanisms, a histogram-based federated algorithm trains each single tree in a distributed manner. At each training round, clients compute and send histograms with gradient and Hessian values to the server, as depicted in Figure 4. The server aggregates the histograms by summing them across all clients and returns the resulting global histograms. This global view enables each client to determine the best next split in the tree based on the combined data distribution. This is possible due to the additive properties of gradients and Hessians. Specifically, as discussed in Section 2.1, each tree node maintains an aggregated gradient G and Hessian H, computed by summing the individual gradients and Hessians of all samples that fall into that node. Therefore, when histograms are aggregated across clients, it is equivalent to increasing the number of samples contributing to the statistics of each node.

FedTree [33]. FedTree provides a custom GBDT implementation. Like NVFlare, it employs histogram aggregation to train tree ensembles in a privacy-preserving manner. However, the key distinction is that in FedTree, the server does not send the aggregated histograms back to the clients. Instead, it uses the aggregated histograms to compute the node information, which is then sent to the clients. This information also includes the aggregated gradient G and Hessian H values. In addition, the authors state that the framework offers three levels of protection. The first level, L_0, provides no protection. The second level, L_1, protects local histograms by using *secure aggregation* [5] for horizontal FL and HE for vertical FL. Finally, the highest level, L_2, applies DP to prevent information leakage from histograms at both the server and client sides.

3 Related Work

The research community has put significant effort into studying adversarial ML. With the spread of FL, new threats have emerged [18, 29, 36]. Specifically, FL systems can be vulnerable to several types of attacks, including free-rider attacks [19], in which participants do not contribute to the training process but still benefit from the global model; utility attacks [3, 51], where adversaries manipulate the training process to gain an advantage; and privacy attacks [46], where sensitive information about the underlying training dataset is at risk. In this work, we focus on privacy attacks and their countermeasures. Therefore, the following describes the main related privacy attacks and defenses.

Attacks. Several prior works focused on privacy attacks in FL, specifically targeting federated Artificial Neural Networks (ANNs). Plain gradient sharing has been proven to leak sensitive information about the training data [42, 68]. Furthermore, different works have improved the effectiveness of gradient inversion attacks in various scenarios. For instance, such attacks can be performed in the presence of gradient compression [63], when the adversary does not have access to the model parameters [66], or to reduce reconstruction complexity [60]. More recently, J. C. Zhao et al. introduced Loki [67], an attack that breaks the anonymity of model aggregation and is effective against secure aggregation. Another type of attack leverages a Generative Adversarial Network (GAN) to generate synthetic data that mimics the training dataset [26]. This approach uses the global model as a discriminator to differentiate between real and synthetic data. The attacker then injects synthetic samples into the training process, gradually refining the GAN to produce increasingly realistic data. Z. Wang et al. [57] propose an improvement to the GAN attack, where the adversary is also able to extract user-specific private information without interfering with the training process. A further improvement is presented in GRNN [44], where the attacker can recover private information from shared gradients without the need for class labels. In the same scenario, H. Wu et al. introduce a class-property inference attack [58], where the adversary aims to infer properties of a specific class in the dataset, focusing on tabular data.

In the tabular data domain, the authors of TabLeak [54] introduce a robust evaluation method for data reconstruction, which we leverage in our study, as it specifically targets tabular datasets.

While most privacy FL attacks have been studied in the context of ANN, much less attention has been given to tree-based models. Some works have explored privacy attacks in vertical FL for tree-based models [12, 38, 52]. However, horizontal settings remain unexplored. Therefore, our work aims to close this gap.

Defenses. Several defense mechanisms have been proposed to enhance privacy in FL systems. Hardware-based solutions, such as secure enclaves [30], provide isolated execution environments that protect computations on the server side. Techniques such as Homomorphic Encryption (HE) [13, 15, 37] offer strong theoretical privacy guarantees by enabling model aggregation on encrypted data. In FL, HE is used to protect client updates from server-side privacy threats. However, its practicality is limited due to the restricted set of operations that can be performed on ciphertexts. Similarly, Secure Multi-Party Computation (MPC) [34, 59] allows multiple parties to jointly compute functions without revealing their inputs.

In FL, the most widely adopted class of MPC is *secure aggregation* [5], which targets server-side threats. Nonetheless, these cryptographic approaches often suffer from substantial computational and communication overhead [2, 32]. Alternative strategies—such as Differential Privacy (DP) [33] and Locality-Sensitive Hashing (LSH) [32]—seek to protect sensitive information by reducing the granularity of shared data. For instance, DP injects noise into model updates to protect against privacy threats on both the client and server sides. Despite their promise, these non-cryptographic defenses introduce trade-offs between privacy and utility, as the noise added for protection can degrade model performance [31]. Among the previously discussed approaches, those currently implemented in FL tree-based systems include Differential Privacy, Secure Multi-Party Computation (via secure aggregation), and Homomorphic Encryption. DP protects against both client- and server-side privacy threats, whereas MPC and HE offer protection only against server-side threats. Regarding frameworks, NVFLARE supports secure aggregation, while FedTree implements all three approaches.

3.1 Research Gap and Motivation

Federated Learning has been extensively studied in the context of Artificial Neural Networks (ANNs), with several works proposing attacks and corresponding countermeasures to address privacy concerns. However, to the best of our knowledge, no prior work has demonstrated a dataset reconstruction attack targeting horizontal federated tree-based systems. This lack reveals a critical gap in the security analysis of federated gradient boosting models, despite their growing use in privacy-sensitive applications.

Tree-based FL frameworks differ substantially from their ANN-based counterparts in terms of model structure, aggregation mechanisms, and information leakage patterns. While adversarial research on ANNs has inspired the development of privacy-preserving techniques, tree-based FL systems remain largely untested against realistic adversarial scenarios. To address this gap, we propose a novel dataset reconstruction attack that exploits the inherent properties of boosted DTs and how they are federated in current FL frameworks.

Importantly, our threat model—discussed in Section 4—assumes that the attacker controls a client. Consequently, the attack can be executed by exploiting only client-side information. As previously explained, privacy defenses such as Homomorphic Encryption and Secure Multi-Party Computation are designed to prevent server-side privacy leakage and, therefore, do not alter the information received by clients. As a result, they are ineffective in our threat model. Moreover, applicable defenses such as Differential Privacy—specifically in its Local Differential Privacy [31] form—have not yet been evaluated against real-world attacks in horizontal FL scenarios. To address this gap, we evaluate the effectiveness of DP against TimberStrike.

In summary, by introducing a novel attack, we aim to systematically uncover key vulnerabilities in tree-based FL systems that allow adversaries to reconstruct private training datasets with measurable accuracy and to evaluate whether existing defenses can mitigate such attacks. This work is further motivated by the goal of proposing theoretical modifications to current frameworks, ultimately paving the way for more robust and privacy-preserving gradient boosting FL architectures.

4 Threat Model

This section defines the threat model by defining the adversary's capabilities, objectives, and target.

Adversary's Capabilities. We assume the adversary can control a client involved in a horizontal FL system where the server is a *trusted entity*. Consequently, they have full white-box access to the trained global model, allowing them to inspect its internal structure. In particular, they can access all the information the clients have at training time. Such information depends on the considered implementation but, in general, includes the trained DTs, their splits, leaf values, hyperparameters, and the aggregated gradient and Hessian values used during training (all definitions about tree-related concepts in Section 2.1). The adversary is modeled as an honest-but-curious participant, as in prior work [54], meaning they strictly follow the protocol without attempting to disrupt the system. However, they seek to infer sensitive information about other participants' training data by analyzing model parameters and training statistics available on the client side. Unlike prior works, which assume the honest-but-curious adversary is located on the server, we assume it resides on one of the clients. This scenario is particularly relevant in FL settings where participants can be market competitors and training data is a valuable asset. Moreover, because the adversary behaves according to the protocol, they are harder to detect—the attack can be conducted entirely "offline."

Adversary's Objective. The adversary aims to reconstruct the training dataset from other participants in the FL process by analyzing the model parameters and training statistics exchanged during training. The goal is to infer sensitive information about the training data, such as the presence of specific samples or the distribution of features and labels in the training data. This information could be used to gain a competitive advantage or to compromise the privacy of the participants.

Targeted Implementations. The adversary targets other clients' datasets by exploiting vulnerabilities in 5 main implementations of federated tree-based systems, including Flower Bagging and Cyclic [4], NVFlare histogram-based [48], FedXGBllr [39], and FedTree [33]. These implementations represent diverse approaches, each varying in the amount of information exposed during training. We consider them representative of the SotA in federated tree-based models and relevant for practical deployments.

5 TimberStrike: Dataset Reconstruction Attack

We propose a novel dataset reconstruction attack, called TimberStrike, to assess the privacy risks associated with training federated tree-based models. TimberStrike specifically targets the implementations of XGBoost and GBDT in horizontal FL settings.

The attack reconstructs clients' training data by exploiting information accessible to an honest-but-curious adversary. As defined in Section 4, our threat model—guided by existing FL implementations—assumes that the adversary, by controlling a client within the system, can access the trained model(s) in plaintext. Consequently, the adversary can inspect tree structures, aggregated gradients, and Hessians. Additionally, they can observe model parameters such as the base score and learning rate, which remain accessible during training. Since these statistics and parameters are fundamental to

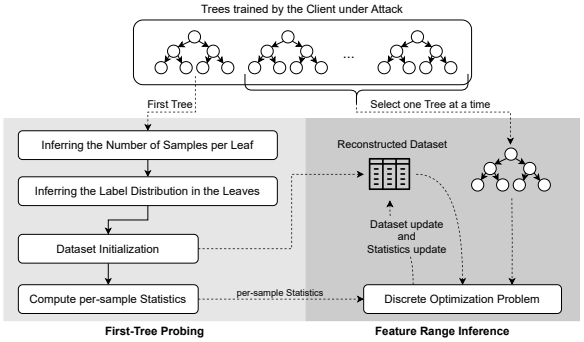


Figure 5: TimberStrike attack schema.

understanding the internals of TimberStrike, we provide all relevant definitions related to gradient boosting in Section 2.1.

The intuition behind our approach is that an attacker, by controlling a client, can exploit the sequential nature of boosted trees. By analyzing the decision splits and gradient statistics shared by other clients, the attacker can infer information about the training data that influenced the tree construction. The attack can be performed at each round of training. However, the more rounds that have been completed, the more trees have been trained. Therefore, the attack will be more accurate. As depicted in Figure 5, TimberStrike can be divided into two main phases that need to be repeated for each client the adversary targets. The first phase, *First-Tree Probing*, examines the first tree trained by the client under attack to infer sample counts and label distributions, and to initialize the reconstructed dataset based on the observed tree splits. The second phase, *Feature Range Inference*, refines the reconstruction by solving an optimization problem on each subsequent tree trained by the client under attack. Following, we provide a detailed explanation of the attack by describing each phase in depth. Finally, we discuss its adaptation to different federated gradient boosting implementations. Note that the attack formalization pertains to a *binary classification* task. However, it can be extended to *multi-class classification*, for which we provide a formalization in Section E.

5.1 First-Tree Probing

As shown in Section 2.3, regardless of the specific federated implementation, the client receives the global model from the Parameter Server (PS) at least once. This means the controlled client has visibility into the global model, which contains trees contributed by all other clients. In this phase, the attacker targets a specific client and selects the *first tree* trained by that client. The motivation for selecting the first tree is that it is built on predictions obtained using the base score. As we will explain, this plays a crucial role in TimberStrike. As depicted in Figure 5, this first phase of TimberStrike can be further divided into four steps, aimed at inferring the number of samples per leaf, inferring the label distribution in the leaves, initializing the dataset, and computing per-sample statistics.

Inferring the Number of Samples per Leaf. After clients receive the aggregated trees from the PS, the adversary can infer the number of samples assigned to each leaf j by analyzing the first tree trained by the chosen victim. Before discussing how we can obtain this information, let us first analyze the gradient and Hessian statistics for a binary classification task. In particular, given a log loss (or binary cross-entropy loss) function, these statistics can be obtained [11, 21] for each sample i, starting from Equation (1):

$$g_i = p_i - y_i, \quad h_i = p_i \cdot (1 - p_i), \tag{3}$$

where the probability score $p_i = \sigma(x)$, x is the sum of outputs of the previously trained trees, $\sigma(x) = \frac{1}{1+e^{-x}}$, and y_i is the label of the i-th sample. Now, by analyzing the first tree, we know that p_i only depends on the base score, which is the constant global bias of our model. Considering base score is already a probability, for each sample during the training of the first tree, we have $p_i = \sigma(\sigma^{-1}(base\_score)) = base\_score$. We can, therefore, write:

$$h_i = base\_score \cdot (1 - base\_score), \tag{4}$$

and the total aggregated Hessian for a given leaf j as:

$$H_j = \sum_{i=1}^{N_j} h_{ij} = N_j \cdot base\_score \cdot (1 - base\_score), \tag{5}$$

where N_j represents the number of samples assigned to the leaf j. Solving for N_j, we obtain:

$$N_j = \frac{H_j}{base\_score \cdot (1 - base\_score)}, \tag{6}$$

By disposing of both the base score and the total aggregated Hessian for each leaf in the tree, we can solve the above equation.

Inferring the Label Distribution in the Leaves. Once TimberStrike infers the number of samples per leaf, it can infer the *distribution of labels* within each leaf, forming the foundation for *dataset initialization*. Indeed, by knowing the label distribution and the number of samples, an adversary can initialize a dataset with the same size and distribution as the targeted one. To obtain this information, we can exploit the aggregated gradient for each leaf j. In gradient boosting, this gradient is given by the formula:

$$G_j = -\frac{leaf\_value_j}{\eta} \cdot (H_j + \lambda), \tag{7}$$

where η is the learning rate and λ is the regularization parameter. Since the gradient for each sample depends on its label, we leverage the following expression to differentiate between samples labeled as 0 and 1 (binary classification):

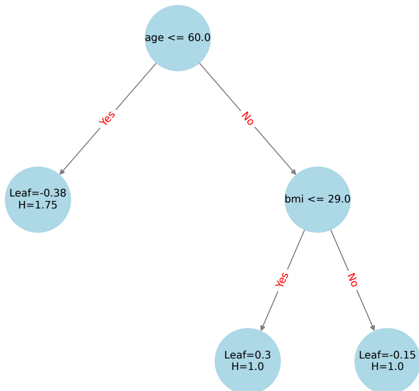
$$G_j = \sum_{i=1}^{N_j} g_{ij} = N_j^{(0)} \cdot base\_score - N_j^{(1)} \cdot (1 - base\_score), \tag{8}$$

where $N_j^{(0)}$ and $N_j^{(1)}$ denote the number of samples with labels 0 and 1 in the leaf j. Given that the total number of samples in the leaf satisfies $N_j^{(0)} + N_j^{(1)} = N_j$, we can write:

$$N_j^{(1)} = N_j \cdot base\_score - G_j, \quad N_j^{(0)} = N_j - N_j^{(1)}, \tag{9}$$

and, disposing of both G for each leaf and the base score, the adversary can solve the above equations to finally compute the label distribution within a certain leaf.

Dataset Initialization. At this stage, the adversary knows both the number of samples per leaf and their label distributions. Therefore, they can generate the exact number of samples assigned from the training set to that leaf, with the exact label distribution. Moreover, in tree-based models, each sample reaches a leaf by following a path dictated by its feature values. TimberStrike exploits this property to



Age	BMI	Per-sample Statistics	Label
$(-\infty, 60.0)$	-	$p = 0.41,\ g = 0.41,\ h = 0.24$	0
$(-\infty, 60.0)$	-	$p = 0.41,\ g = 0.41,\ h = 0.24$	0
$(-\infty, 60.0)$	-	$p = 0.41,\ g = 0.41,\ h = 0.24$	0
$(-\infty, 60.0)$	-	$p = 0.41,\ g = 0.41,\ h = 0.24$	0
$(-\infty, 60.0)$	-	$p = 0.41,\ g = 0.41,\ h = 0.24$	0
$(-\infty, 60.0)$	-	$p = 0.41,\ g = 0.41,\ h = 0.24$	0
$(-\infty, 60.0)$	-	$p = 0.41,\ g = 0.41,\ h = 0.24$	0
$(60.0, \infty)$	$(-\infty, 29.0)$	$p = 0.57,\ g = -0.43,\ h = 0.24$	1
$(60.0, \infty)$	$(-\infty, 29.0)$	$p = 0.57,\ g = -0.43,\ h = 0.24$	1
$(60.0, \infty)$	$(-\infty, 29.0)$	$p = 0.57,\ g = -0.43,\ h = 0.24$	1
$(60.0, \infty)$	$(-\infty, 29.0)$	$p = 0.57,\ g = -0.43,\ h = 0.24$	1
$(60.0, \infty)$	$(29.0, \infty)$	$p = 0.46,\ g = -0.54,\ h = 0.25$	1
$(60.0, \infty)$	$(29.0, \infty)$	$p = 0.46,\ g = 0.46,\ h = 0.25$	0
$(60.0, \infty)$	$(29.0, \infty)$	$p = 0.46,\ g = 0.46,\ h = 0.25$	0
$(60.0, \infty)$	$(29.0, \infty)$	$p = 0.46,\ g = 0.46,\ h = 0.25$	0

Figure 6: Decision Tree (DT) with its reconstructed dataset initialized after the First-Tree Probing phase. In this example, we use $\lambda = 1$, $learning\_rate = 0.3$, and $base\_score = 0.5$.

impose constraints on the range of features in a sample through the observed leaf assignments. In summary, at this step, the adversary can generate the first instance of the reconstructed dataset (an example in Figure 6).

Compute per-sample Statistics. This final step is essential for executing the second phase and aims to complete each reconstructed sample with three statistics used during training by gradient boosting algorithms. In particular, for each generated sample i, we compute the gradient g_i and the Hessian h_i according to Equation (3), using the probability score p_i computed as:

$$p_i = \sigma(\sigma^{-1}(base\_score) + \sum_{t=1}^{T} leaf\_score_i^{(t)}). \qquad (10)$$

where $\sigma^{-1}(x) = \log\left(\frac{x}{1-x}\right)$, T is the number of already analyzed trees (in the first phase $T = 1$), and $leaf\_score_i^{(t)}$ is the prediction value assigned to the leaf of the tree t in which the i-th sample falls.

5.2 Feature Range Inference

Once the initial feature ranges have been inferred, TimberStrike refines the generated dataset by formulating a Mixed-Integer Linear Programming (MILP) problem to determine the precise leaf assignment for each sample in each subsequent tree. In other words, after analyzing the first tree, TimberStrike analyzes and solves a MILP problem for each other tree trained by the client under attack. Since each sample follows a unique path through the tree, based on its feature values, we leverage the feature constraints accumulated so far to restrict the possible set of leaves a sample can reach. This defines a subset of all possible leaves, which we use as constraints in our optimization problem, which minimizes the discrepancy between reconstructed and original aggregated gradients and Hessians, progressively refining feature estimates.

Looking at a tree in the ensemble, for each sample, we define a binary assignment variable x_{ij}, which indicates whether sample i is assigned to leaf j. The constraints of the optimization problem enforce that each sample is assigned to exactly one leaf and that this leaf belongs to the set of reachable leaves based on previously inferred feature ranges. As for any optimization problem, we need to define the *constants*, the *variables*, the *constraints*, and the *objective function*. Following the formalization of the designed problem.

Constants. We define the following constants:

- $I = \{1, \ldots, n\}$: set of training samples.
- $J = \{1, \ldots, m\}$: set of leaves.
- G_j: aggregated gradient of leaf j.
- H_j: aggregated Hessian of leaf j.
- p_i: probability score of sample i from previous trees.
- g_i: gradient of sample i, computed from p_i and its label.
- h_i: Hessian of sample i, computed from p_i and its label.
- $L_i \subseteq J$: set of leaves that sample i can reach, based on inferred feature constraints.

Variables. We define the following variables:

- $x_{ij} \in \{0, 1\}$: binary variable indicating whether sample i is assigned to leaf j.

Constraints. Each sample must be assigned to exactly one leaf:

$$\sum_{j \in J} x_{ij} = 1, \quad \forall i \in I. \qquad (11)$$

The leaf assignments must respect the inferred feature constraints:

$$x_{ij} = 0, \quad \forall i \in I, \forall j \notin L_i. \qquad (12)$$

Objective Function. The objective is to minimize the discrepancy between the reconstructed and original gradients and Hessians:

$$\min \sum_{j \in J} (\sum_{i \in I} x_{ij} \cdot g_i - G_j)^2 + (\sum_{i \in I} x_{ij} \cdot h_i - H_j)^2. \qquad (13)$$

By iteratively solving the above optimization problem, we update the feature ranges for each sample and the gradient and Hessian values by re-executing the last step of the previous phase (Compute per-sample Statistics). Following this process, we progressively refine the estimated feature values of each sample. At the end of the process, TimberStrike achieves a reconstruction of the training data that is consistent with the observed model.

5.3 TimberStrike Details

We now present how our attack can be adapted to target each implementation discussed in Section 2.3 (Flower Bagging and Cyclic [4], NVFlare histogram-based [48], FedXGBllr [39], and FedTree [33]). In addition, we discuss how these implementations impact the reconstruction granularity. Specifically, while Flower Bagging, Cyclic, and FedXGBllr allow for targeting and reconstructing *local* clients' datasets, histogram-based systems such as NVFlare and FedTree

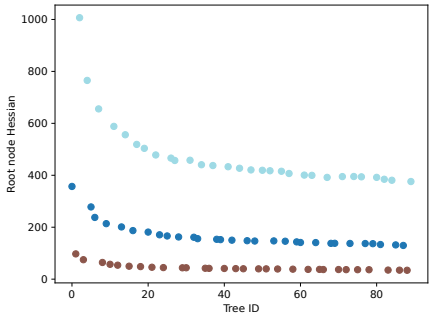


Figure 7: Root node Hessian for 90 trees trained using the Flower XGBoost Bagging implementation. Each client trains 30 trees, which are then identified under the same client by minimizing the distance with the trees in the previous round.

only enable reconstruction of the aggregated *global* dataset—i.e., a union of all clients' datasets. We adapt TimberStrike accordingly, detailing the differences in the attack implementation for each case.

5.3.1 Flower XGBoost Bagging and Cyclic. In these two implementations, the first tree in the model is not necessarily trained by the client under attack. Indeed, the Parameter Server (PS) concatenates the trees in arrival order. Furthermore, in both implementations, each client trains local trees using models from previous rounds, which incorporate trees trained by other clients. As a result, the aggregated gradient and Hessian statistics at each round are computed on samples that do not necessarily belong to the client under attack. Therefore, at each round, we infer feature ranges for each sample and perform a weighted average of the leaf values from trees not trained by the client under attack, using the Hessian values as weights. Consequently, we approximate the true leaf in which the sample is likely to fall across the trees of other clients while preserving the feature range constraints that are relevant only to trees trained by the victim client. Finally, looking only at the trees trained by the victim client, we proceed as usual by solving the optimization problem.

Flower XGBoost Bagging differs from the Cyclic variant in that client ordering is not fixed; instead, it depends on client arrival times or follows a randomized schedule. Therefore, to recognize the trees from the same client, we look at the Hessian and gradient statistics. In particular, the idea is that, given the data on which a single client trains a tree are always the same across rounds, it is possible to find a relation between a tree at round r and a tree trained by the same client at round $r-1$ by searching for the minimum distance between the hessian values in a tree at round r and a tree at round $r-1$. Still, this mechanism only affects the second phase. In case of issues in reconstructing the chain of trees trained by the same client, the adversary can still use the *First-Tree Probing* phase alone using only the first received trees. In Figure 7 we plot the Hessian values of the root nodes for an example using 3 clients. Moreover, an honest-but-curious adversary lacks direct knowledge of the mapping between tree ordering and client identities, but while it is not possible to attribute reconstructed samples to specific clients, the adversary can still infer that certain samples originate from the same client.

5.3.2 FedXGBllr. For such an implementation, the attack doesn't need to be adapted. We extract the victim client's local trees and apply the method described in Section 5. The first round of the protocol, where the trees contained in the XGBoost models are shared, is enough to reconstruct the entire training set. The attack enables user-level reconstruction, as the models are trained separately on clients' data, and the locally trained trees are shared with all other clients (including the adversary) in the initialization phase.

5.3.3 NVFlare and FedTree. These two implementations aggregate histograms to determine, at each round, the best splits that each client should perform during the training of a single tree. As a result, each client receives only aggregated gradient and Hessian statistics derived from the combined histograms (i.e., the sum of the clients' histograms). The attack proceeds as described in the general formalization. However, in this case, the adversary can only reconstruct the *global* dataset (i.e., the union of all clients' datasets), since these systems expose only aggregated model updates rather than individual local trees. Consequently, the attacker observes the boosting of *global* trees.

6 Experimental Evaluation

Our experimental evaluation aims to test TimberStrike to address the following research questions:
RQ1: Is our attack effective across all considered tree-based FL systems? Does the *Feature Range Inference* phase enhance effectiveness, and what is the impact of the *First-Tree Probing* phase alone?
RQ2: How does tree depth influence the effectiveness of the attack in the considered systems? Specifically, does increased depth and potential overfitting help the dataset reconstruction?
RQ3: Can the attack's effectiveness be mitigated by applying classical defenses such as DP during training?

6.1 Experimental Setup

We evaluate TimberStrike on five different federated systems—four based on XGBoost and one on GBDT. In particular, we adapt and evaluate our attack on XGBoost Bagging, Cyclic, and FedXGBllr [39], all implemented in Flower [4]. Additionally, we apply TimberStrike to histogram-based implementations in NVFlare [48] and FedTree [33]. We consider a general scenario with 3 clients collaboratively training a federated gradient boosting model composed of 100 trees with a depth d from 3 to 8 (we perform 6 runs for each implementation by increasing d). To evaluate the scalability of TimberStrike, in Section 6.2.4 we vary the number of clients up to 30. One client acts as an honest-but-curious adversary attempting to extract sensitive information from the other clients performing the TimberStrike attack. In the Flower implementations, where TimberStrike can target a specific client, we designate one of the two remaining clients as the victim. In histogram-based implementations, since the attack can only reconstruct the global dataset (i.e., the union of all clients' datasets), we compare the reconstructed dataset with the original dataset, which consists of the combined datasets from all clients. From now on, we will refer to the reconstruction targeting a specific client as *local reconstruction*, while the reconstruction of the global dataset, as in the case of histogram-based implementations, will be referred to as *global reconstruction*. Finally, to limit the computational load of the Mixed-Integer Linear

Table 1: F1-score, and Area Under the ROC Curve (AUC-ROC) on the test set for the binary classification task on the Stroke Prediction and Pima Indians Diabetes datasets.

Dataset	Implementation	Depth 3		Depth 4		Depth 5		Depth 6		Depth 7		Depth 8	
		F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
Stroke	FedXGBllr	0.215	0.635	0.218	0.628	0.183	0.620	0.157	0.617	0.105	0.608	0.097	0.614
	Flower XGBoost Bagging	0.213	0.755	0.215	0.750	0.185	0.750	0.176	0.746	0.138	0.731	0.160	0.751
	Flower XGBoost Cyclic	0.237	0.748	0.157	0.738	0.158	0.731	0.162	0.760	0.176	0.734	0.158	0.709
	NVFlare	0.165	0.767	0.203	0.774	0.203	0.767	0.164	0.775	0.103	0.744	0.128	0.761
	FedTree	0.207	0.766	0.200	0.764	0.209	0.766	0.110	0.767	0.130	0.768	0.108	0.769
Pima	FedXGBllr	0.587	0.816	0.603	0.806	0.572	0.810	0.531	0.806	0.607	0.809	0.582	0.814
	Flower XGBoost Bagging	0.638	0.769	0.587	0.743	0.577	0.716	0.506	0.689	0.537	0.703	0.585	0.718
	Flower XGBoost Cyclic	0.573	0.753	0.506	0.669	0.490	0.669	0.591	0.755	0.549	0.719	0.529	0.679
	NVFlare	0.643	0.817	0.676	0.834	0.639	0.821	0.636	0.831	0.627	0.832	0.660	0.825
	FedTree	0.586	0.810	0.596	0.826	0.606	0.798	0.604	0.805	0.604	0.806	0.610	0.815

Table 2: Tolerance ϵ for the Reconstruction Accuracy (RA) on the top 5 features of each of the datasets.

Top 5	**Avg. Glucose**	**Work**	**Residence**	**Heart disease**	**Label**
Stroke	17.03	cat*	cat*	cat*	cat*
Top 5	**BMI**	**Diabetes pedigree**	**Glucose**	**Age**	**Label**
Pima	2.50	0.105	10.06	3.77	cat*

*categorical, the predicted feature needs to match perfectly with the original.

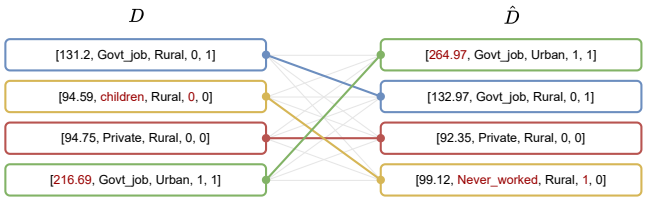


Figure 8: Reconstruction assessment of a 4-sample dataset, using the 5 most important features from Stroke. Each sample in the reconstructed dataset $\hat{D}$ is compared with one in the original dataset D. In red, we highlight the wrong reconstructed values. Here, we are able to fully reconstruct the first and third samples (blue and red), the 60% of the second sample (yellow), and the 80% of the last sample (green). This leads to an RA of 85%.

Programming problems in our *Feature Range Inference* phase (more details in Section D), we impose a time constraint of 10 minutes for each tree analyzed.

Hardware and Software Testing Environment. We conduct our experiments on a machine equipped with an Intel 11th Gen Core i7-1165G7 processor (8 cores, 16 threads, 2.80 GHz base clock, 4.70 GHz max turbo) and 16 GB of RAM. The software environment includes Python 3.10, XGBoost 2.1.0, and the latest version of Gurobi [22].

Datasets. We conduct our experiments using two publicly available binary classification datasets from the healthcare domain, particularly renowned for the privacy of the training data: the *Stroke Prediction Dataset* [24] and the *Pima Indians Diabetes Dataset* [14]. The Stroke Prediction Dataset consists of 5110 records, each representing an individual with attributes related to stroke risk factors. It includes 11 features (including the stroke column), categorized into demographic features (*age, gender, marital status, residence type*), medical conditions (*hypertension, heart disease*), lifestyle factors (*smoking status, work type*), and clinical measurements (*average glucose level, BMI*). In summary, it contains 3 numerical and 7 categorical features. This dataset is highly imbalanced, with only 4.87% of samples corresponding to stroke cases. Therefore, we apply SMOTE-NC [9] to balance the classes. The Pima Indians Diabetes Dataset contains 728 records and 9 features (including the outcome column). The features include *pregnancies, OGTT (Oral Glucose Tolerance Test), blood pressure, skin thickness, insulin, BMI, age*, and *pedigree diabetes function*. All these features are numerical.

Data Distribution. We focus on a scenario where the distribution of data across clients is non-IID, as it better reflects real-world conditions. As with any other work in the SotA, to simulate the non-IID case, we use the Dirichlet distribution [50] ($\alpha = 0.3$) to partition the data across the clients.

Evaluation Metrics. We evaluate the effectiveness of our approach using three main metrics: the *Reconstruction Accuracy (RA)* [54], the *F1-score*, and the *Area Under the ROC Curve (AUC-ROC)*. They are used to assess the efficacy of the reconstruction attack (Reconstruction Accuracy) and the performance of the model on the trained task (F1 and AUC-ROC). While the F1-score and the AUC-ROC are well-known classification metrics, the Reconstruction Accuracy is a metric introduced in TabLeak [54], as it is the first work on tabular data leakage. The RA of a reconstructed dataset $\hat{D}$ is computed by averaging the reconstruction accuracy scores of all its samples. This score $ra_{\hat{x}}$ for a single sample $\hat{x} \in \hat{D}$ is defined as the proportion of features for which the inferred value (or range) overlaps, within a specified tolerance, with the corresponding ground-truth value in a paired sample $x \in D$, where D is the original dataset. Formally, $ra_{\hat{x}}$ is computed as follows:

$$ra_{\hat{x}}(x,\hat{x}) = \frac{1}{K+L}\left(\sum_{i=1}^{K}\mathbb{1}\left(x_i^{(k)} = \hat{x}_i^{(k)}\right) + \sum_{i=1}^{L}\mathbb{1}\left(\hat{x}_i^{(l)} \in \left[x_i^{(l)} - \epsilon_i, x_i^{(l)} + \epsilon_i\right]\right)\right) \quad (14)$$

where K is the number of categorical features, L is the number of continuous features, and ϵ_i is the tolerance for the i-th continuous feature. Each ϵ_i is computed by taking the standard deviation of the i-th continuous feature in the training dataset and multiplying it by a constant. In our experiments, we follow the setup of TabLeak [54], and set $\epsilon_i = 0.319 \cdot \sigma_i^{(l)}$, obtaining the following error distribution:

$$P\left(\mu - 0.319 \cdot \sigma \leq x \leq \mu + 0.319 \cdot \sigma\right) = 2\Phi(0.319) - 1 \approx 25\%, \quad (15)$$

where Φ denotes the cumulative distribution function of the standard normal distribution. The reconstruction assessment requires each reconstructed sample to be compared with its corresponding

ground truth, giving equal weight to all features. To establish a one-to-one correspondence between the reconstructed and original datasets, we employ a bipartite matching algorithm—specifically, the Hungarian method [28]—as done in TabLeak [54]. In Figure 8 we show an example of reconstruction assessment. Finally, to visualize the practical impact of high RAs, Table 2 presents the tolerance required for a feature in the reconstructed sample to be considered a match with its corresponding original sample.

Binary Classification task performance. To observe the relationship between overfitting/underfitting and RA, we show in Table 1 how the models perform for each considered dataset, federated gradient boosting implementation, and tree depth. The results indicate that, for the Stroke dataset, the model achieves its best F1-score and AUC-ROC with lower tree depth. This suggests that increasing the tree depth leads to overfitting. In contrast, this trend is not observed for the Pima dataset, which is significantly smaller.

Defenses. We also evaluate TimberStrike in a scenario where clients use Differential Privacy (DP) to protect their updates from potential privacy leakage. As discussed in Section 3.1, to the best of our knowledge, this is the only existing method that theoretically mitigates our attack under the given threat model. The authors of FedTree [33] introduce three layers of privacy protection (see Section 2.3), which represent the current SotA in federated tree-based systems. However, since our attack reconstructs a dataset from the client side, neither *secure aggregation* nor HE (protection level L_1) mitigates TimberStrike, as they only protect clients' updates from server-side privacy threats. Therefore, we evaluate their L_2 protection, which incorporates DP, by varying different *privacy budget values* (ϵ) (more details on ϵ-DP and its implementation in FedTree are provided in Section B). The experiments with DP pertain only to FedTree, as it is the only framework that implements it for tree-based models across the considered frameworks, to the best of our knowledge. However, since their current implementation is complete only in the vertical FL scenario, we extend it to the horizontal FL setting to the best of our ability.

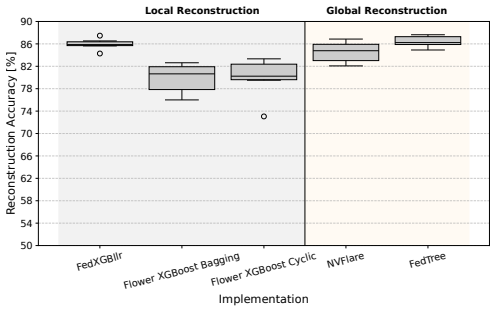
6.2 TimberStrike Effectiveness (RQ1 & RQ2)

We aim to investigate the overall performance of TimberStrike, the contribution of its two phases, and how hyperparameters such as tree depth influence the reconstruction performed by our attack. Following, we present the results using aggregated plots, while more detailed tables with raw results are provided in Section C. Note that, except for the plots used to show the impact of the client size on TimberStrike and the impact of tree depth on TimberStrike, all other plots comparing the different implementations in terms of Reconstruction Accuracy distinguish between *local* and *global* reconstruction, as discussed in Section 6.1. In particular, local reconstruction refers to implementations that enable targeting a specific client, while global reconstruction refers to those that allow only the reconstruction of the global dataset.

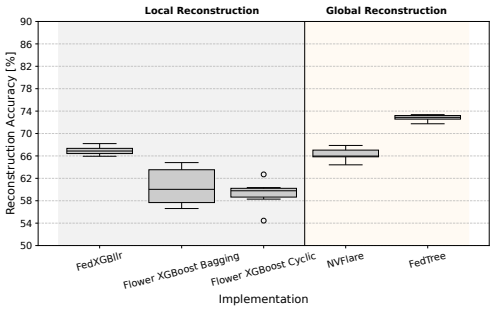
6.2.1 Attack performance on different implementations. We present the boxplots of the Reconstruction Accuracy (RA) (in percentage) on both the Stroke (Figure 9a) and Pima (Figure 9b) datasets, corresponding to the six runs performed on each system while varying the depth of the trees. Note that the results account for all features in the dataset—11 for the Stroke dataset and 9 for the Pima dataset. The general performance of TimberStrike shows a minimum of 73.05% RA on Stroke and 54.45% RA on Pima, both registered on Flower XGBoost Cyclic, while achieving a maximum of 86.86% RA on Stroke (NVFlare) and 73.37% RA on Pima (FedTree), demonstrating that a significant amount of the original dataset can be reconstructed within a tolerance in both datasets. These results are in line with the SotA of RA on ANNs for tabular datasets [54]. However, the RA on the Pima dataset is worse than what TimberStrike obtains on the Stroke one. This discrepancy is due to differences in dataset sizes and the composition of the dataset, which contains only numerical features. This result indicates that reconstructing categorical features is easier than reconstructing numerical ones, a finding also observed by the TabLeak authors. The two datasets exhibit similar behavior when comparing different systems. In both datasets, the implementations where TimberStrike achieves the highest and least skewed RA are the two histogram-based systems (NVFlare and FedTree) and FedXGBllr. The results obtained with the histogram-based implementations demonstrate that, even if we globally reconstruct the dataset, TimberStrike can still achieve high performance. Additionally, FedXGBllr confirms what its implementation suggests: the plain exposure of trees significantly increases the risk of privacy leakage. In contrast, both the Cyclic and Bagging implementations in Flower prove to be more resistant to attacks. This can be attributed to their interleaved nature, where each client continuously trains a new tree based on the trees received from other clients.

6.2.2 Impact of the feature importance on the RA. The results already presented demonstrate that TimberStrike can effectively reconstruct clients' data with a high accuracy. However, we argue that the importance of features in predictions significantly impacts the RA. If this holds, it implies that an adversary may achieve even better reconstruction performances on the most relevant features, and may also identify which features they can "trust" more in the reconstructed dataset. To validate this intuition, we evaluate the Reconstruction Accuracy on the top features (i.e., the most important ones) and compare it to the RA achieved on the full dataset. Specifically, we train a centralized XGBoost model, extract the feature importance, and select the top five features (including the label) for each dataset, visible in Table 2. In Figures 9c and 9d, we compare the RA evaluated on all features with the one computed using only the top five features. As the figure illustrates, feature importance significantly impacts TimberStrike. Indeed, in the best-case scenario, TimberStrike achieves an RA of over 95.63% for FedXGBllr on the Stroke dataset. Furthermore, TimberStrike demonstrates that even with models trained on small and numerical datasets like Pima, on the most relevant (and therefore most impactful on the trees) features, it can still reconstruct well the training data, as shown by the substantial increase in RA.

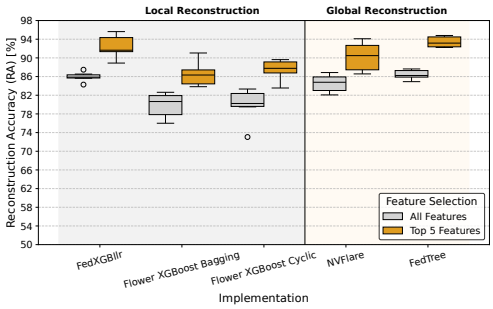
6.2.3 Contribution of the Feature Range Inference phase. In Figures 9e and 9f, we show the RA achieved by TimberStrike after the first phase (First-Tree Probing) and after the complete attack. As the figure illustrates, the second phase (Feature Range Inference) effectively improves the RA compared to the first phase alone. In particular, the Feature Range Inference phase increases the average RA across all systems by 7.19% on the Stroke dataset and by 1.97% on the Pima dataset. However, there are two exceptions (Flower



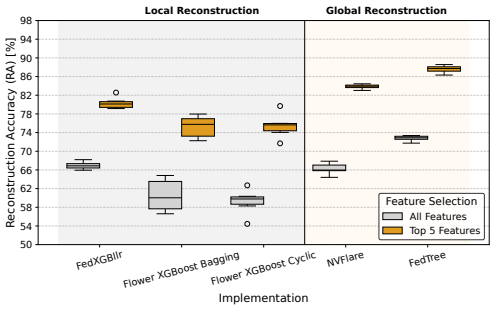
(a) Stroke - Overall Reconstruction Performance



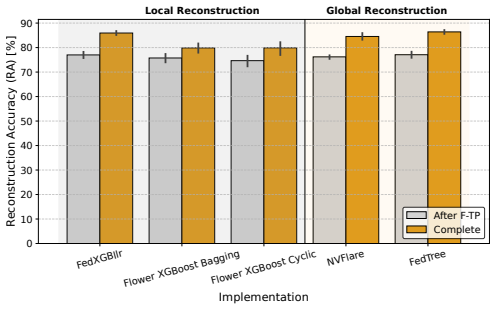
(b) Pima: Overall Reconstruction Performance



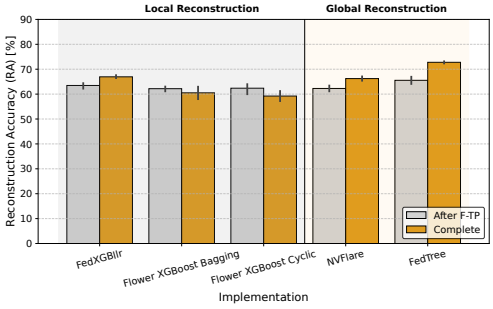
(c) Stroke: Most Relevant Features Performance



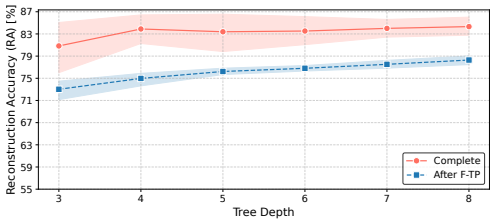
(d) Pima: Most Relevant Features Performance



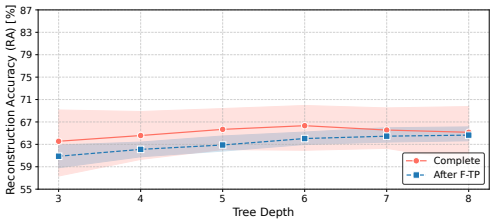
(e) Stroke: First-Tree Probing (F-TP) RA vs Final RA



(f) Pima: First-Tree Probing (F-TP) RA vs Final RA



(g) Stroke: RA with Various Tree Depths



(h) Pima: RA with Various Tree Depths

Figure 9: Experimental results on the Stroke Prediction Dataset (left) and the Pima Indians Diabetes Dataset (right).

Bagging and Cyclic on Pima), where the second phase decreases the average results. Such an exception occurs because the interleaved nature of these implementations, combined with the small dataset size of the Pima dataset, prevents the correct identification of the victim client's tree chain (see Section 5.3). Interestingly, the RA achieved using only the first phase demonstrates that the first tree alone is sufficient to cause significant privacy leakage.

6.2.4 Impact of Client Size on Global Reconstruction. We previously discussed the difference between global and local reconstruction. In the case of the two histogram-based methods, the reconstruction targets the global dataset rather than any single client. Here, we investigate whether increasing the number of participating clients leads to a loss of detail in the aggregated information. To show the impact of client scaling on global reconstruction, we present in Figure 10 the results of an experiment where we evaluate TimberStrike against FedTree on the Stroke Dataset in five different scenarios (3, 5, 10, 20, and 30 clients). As the plot depicts, except for a small drop from the scenario with 3 clients and the scenario with 5 clients, the

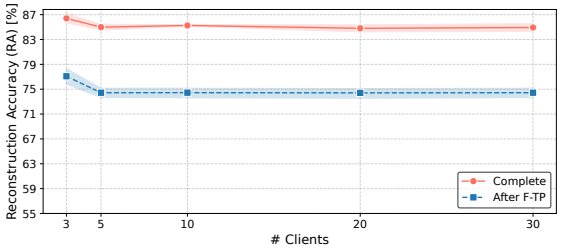


Figure 10: RA on the Stroke dataset using FedTree, with varying numbers of clients involved in the training.

effect of client set size is negligible for the rest of the scenarios, as no significant drop in RA is registered. Therefore, we can state that the RA is poorly affected by the number of clients when performing a global reconstruction on a histogram-based system. The theoretical basis for this result derives from the histogram aggregation mechanism. In horizontal FL, client-side histograms are aggregated by summation. This preserves the characteristics of the individual local histograms, making the resulting global histograms equivalent to the statistics obtained by training a centralized model on the entire dataset. However, as the authors of FedTree [33] point out, when the splits proposed by different clients diverge, they merge the local histograms by approximate summation. Considering that increasing the number of clients leads to a finer partitioning of the dataset, and consequently fewer samples per client, this approximation becomes the source of performance degradation for TimberStrike. However, such degradation is negligible, as shown in Figure 10.

Answer to RQ1. TimberStrike achieves a Reconstruction Accuracy comparable to the SotA of RA on ANNs for tabular datasets. Additionally, its RA significantly improves, reaching a maximum of 95.63%, on the most important features in the dataset. The Feature Range Inference phase increases the average RA across all systems by 7.19% compared to First-Tree Probing on the Stroke dataset and by 1.97% on the Pima dataset. Nonetheless, results demonstrate that First-Tree Probing can achieve good performance even on its own.

6.2.5 Impact of the tree depth. When training a gradient boosting model, it is possible to set the tree depth. This hyperparameter is crucial for training performance, and we expect that it also impacts the RA of our attack. In Figures 9g and 9h, we show the trend of Reconstruction Accuracy, averaged from the five different implementations, as the tree depth used for training in the ensemble varies. As the plots illustrate, the first phase is significantly impacted by the tree depth. This result is expected: deeper trees contain more splits, allowing us to extract more information from the first tree. Furthermore, this result is in line with the SotA of privacy attacks that demonstrate overfitting (demonstrated in Table 1 for Stroke) helps privacy leakage [65]. In contrast, the overall attack results do not exhibit a consistently increasing trend. This can be attributed to the time constraint imposed on solving the MILP problem in the Feature Range Inference phase. Higher tree depth increases complexity and, eventually, performance saturates.

Table 3: Model Utility (F1-score and AUC-ROC), RA after First-Tree Probing (F-TP) and final RA under DP. Both histogram-level and total ϵ are reported.

Features	$\epsilon_{\text{histogram}}$	ϵ_{total}	F1	AUC	RA (F-TP)	RA
All Features	No Defense	–	0.601 ± 0.009	0.810 ± 0.010	65.54 ± 1.95	72.78 ± 0.60
	1	200	0.536 ± 0.036	0.781 ± 0.033	60.65 ± 3.96	70.52 ± 1.77
	0.25	50	0.481 ± 0.043	0.650 ± 0.037	59.97 ± 4.10	61.59 ± 3.22
	0.125	25	0.448 ± 0.042	0.605 ± 0.055	60.26 ± 1.91	54.43 ± 4.63
Top 5 Features	No Defense	–	0.601 ± 0.009	0.810 ± 0.010	82.65 ± 2.03	87.59 ± 0.81
	1	200	0.536 ± 0.036	0.781 ± 0.033	75.95 ± 3.88	85.84 ± 1.17
	0.25	50	0.481 ± 0.043	0.650 ± 0.037	73.26 ± 7.73	75.70 ± 4.17
	0.125	25	0.448 ± 0.042	0.605 ± 0.055	74.18 ± 3.32	67.65 ± 5.54

Answer to RQ2. Increasing tree depth enhances the attack's effectiveness in the initial phase by providing more splits, which allows for greater information extraction. This supports the intuition that overfitting increases privacy leakage risks. However, as depth increases further, the overall attack does not show a consistently improving trend. This is due to the higher computational complexity in the Feature Range Inference phase and the time constraint, which leads to performance saturation.

6.3 DP Effectiveness against TimberStrike (RQ3)

We now want to assess if SotA defenses mitigate TimberStrike. To do so, we test the use of Differential Privacy (DP) as a defense against TimberStrike by using the FedTree implementation.

In this experiment, we use only the Pima dataset, as the Stroke dataset already exhibits poor performance, making it difficult to observe significant changes in model performance. We analyze TimberStrike's behavior by varying a fundamental parameter for DP, i.e., the privacy budget ϵ. In ϵ-DP (more details in Section B), this parameter measures the "privacy loss," meaning that the higher the ϵ, the less privacy the update preserves.

Table 3 depicts the results obtained by evaluating TimberStrike both without defense and with varying values of ϵ, selected according to SotA settings. We report both the *histogram-level* ϵ ($\epsilon_{\text{histogram}}$) and the corresponding *total* ϵ (ϵ_{total}). Note that in FedTree, ϵ-DP is satisfied at the histogram level (details in Section B). Unlike ANN-based federated protocols, here the updates that a client aims to protect are the histograms containing the proposed splits rather than the entire model. Thus, $\epsilon_{\text{histogram}}$ is the effective privacy parameter governing each shared update. ϵ_{total} depends on the number of trees and is provided for completeness, but it does not represent the privacy guarantee for any individual communication round.

Analyzing the model's performance, we observe a significant drop even under the least restrictive ϵ. This decline is particularly evident in the F1-score and persists at lower values of ϵ. Overall, compared to the "no defense" configuration, we observe a maximum average decrease of 0.153 in F1-score and 0.205 in AUC-ROC, highlighting the impact of DP on model usability. On the other hand, when examining RA, we find that the reduction in performance between the "no defense" configuration and the least restrictive privacy setting is less pronounced than the utility drop, with an average decrease of 2.26% when considering all features and 1.75% when considering only the top five. Additionally, the First-Tree Probing phase of our attack demonstrates to be less affected by DP than the overall attack, even showing a higher RA when $\epsilon_{\text{histogram}} = 0.125$.

Overall, while DP reduces the effectiveness of our attack by sacrificing model utility, it does not fully mitigate TimberStrike. Indeed, even under the most aggressive privacy setting and considering the entire feature space, TimberStrike can still achieve an RA > 50%.

Answer to RQ3. Differential Privacy reduces the attack's effectiveness but does not fully mitigate it. Indeed, while DP lowers the Reconstruction Accuracy, it also significantly degrades model utility, with average drops of 0.153 in F1-score and 0.205 in AUC in the most aggressive scenario considered. Moreover, even under the strictest privacy setting, the attack still achieves an RA > 50%, which means DP is insufficient as a standalone defense.

7 TimberStrike Mitigation Guidelines

Our evaluation shows that while DP reduces the effectiveness of TimberStrike, it does not fully mitigate the attack and significantly degrades model utility. Additionally, our SotA analysis suggests that other existing defenses fail to address the threat model assumed for TimberStrike. Among the implementations considered, histogram-based methods appear to be the most promising direction for limiting information accessible to clients, as they only allow the reconstruction of global rather than local datasets.

Motivated by our findings, we argue that a tree-based FL protocol must be designed with a clear understanding of the unique attack surfaces introduced by the federated setting and the model itself. In light of this, we define guidelines for the future design of privacy-preserving federated tree-based systems. Specifically, we describe how histogram-based frameworks like FedTree can be modified to limit client-side visibility of globally aggregated statistics while theoretically preserving the functionality.

TimberStrike exploits FedTree's sharing of first-order and second-order gradients (G and H). While server-side threats can already be mitigated by existing defenses such as HE and secure aggregation, mitigating this vulnerability on the client side requires a different approach. To this end, we propose a theoretical restriction on the information broadcast by the server. In particular, the protocol should ensure that the server transmits only the final split decision (or leaf value) to the clients, while keeping the underlying *global* gradient (G) and Hessian (H) statistics confidential. The training of a node can be refined as follows.

Client side. For each feature a, each client i computes and sends to the server a local histogram $H_a^{(i)}$, which contains the gradients and Hessian values of the split points proposed by the client i.

Server side. Upon receiving $H_a^{(i)}$ from each client i, the server performs the following steps:

(1) Aggregates the statistics for each proposed split point by summing the histograms (as proposed in FedTree [33]):

$$H_a = \sum_{i \in [C]} H_a^{(i)}.$$

(2) Computes the gain $\mathcal{G}(s)$ for each candidate split using the formula in Equation (2) and selects the optimal split:

$$s^* = \arg\max_s \mathcal{G}(s),$$

or, if it is a leaf node, compute the *leaf value*.

(3) Broadcasts only the selected split s^* or *leaf value*.

While we believe that this approach effectively eliminates the information leakage vector exploited by TimberStrike —specifically, access to global gradients and Hessians—we leave a detailed empirical evaluation of its impact on federated learning performance to future work. Moreover, this design may reduce transparency and explainability, as clients no longer have visibility into the rationale behind the split decisions. Investigating this trade-off between privacy and interpretability is also left as future work.

8 Limitations and Future Work

The computational complexity of our attack, particularly during the *Feature Range Inference* phase, is theoretically exponential in the worst-case scenario (see Section D for further details). Although in our experiments this complexity is limited by the imposed time constraint on the optimizer, it still represents a limitation.

Furthermore, as previously discussed, histogram-based mechanisms allow each attacker controlling a client to see only aggregated statistics, thereby limiting TimberStrike. Indeed, with this specific mechanism, TimberStrike can only reconstruct the global training dataset (i.e., the union of all clients' datasets) rather than enabling user-level reconstruction. Additionally, the interleaved nature of the Bagging and Cyclic implementations affects the precision of our reconstruction, as demonstrated by the results.

Our current work represents a first attempt to demonstrate privacy leakage risks in tree-based horizontal FL systems. Looking ahead, important research directions emerge from both our guidelines on a mitigation strategy and the current limitations of our approach. Therefore, we plan to implement a novel tree-based FL system, for which we gave an insight in Section 7.

In addition, future research will explore heuristics to reduce the exponential worst-case complexity and empirically evaluate our attack on tasks beyond binary classification (see Section E for a formalization of the multiclass classification task). Finally, future work could explore how incorporating prior model knowledge affects the success and robustness of adversarial strategies.

9 Conclusion

This work demonstrated that tree-based horizontal FL systems are vulnerable to privacy leakage attacks by introducing TimberStrike, a dataset reconstruction attack. Our attack exploits aggregated statistics and tree splits to recover sensitive data from other clients' training sets. Our evaluation across five different SotA deployed approaches and two tabular datasets from the healthcare domain showed that TimberStrike achieves a Reconstruction Accuracy (RA) comparable to SotA dataset reconstruction methods for ANNs on tabular datasets. Moreover, while applying Differential Privacy during training reduced the RA, it failed to fully neutralize our attack while significantly compromising model utility. The results underscore that federating standard XGBoost implementations inherently expose privacy vulnerabilities. We suggest that future protocols should retain the benefits of histogram aggregation while avoiding the transmission of aggregated statistics to clients. Although our approach provided critical insights, its limitations in computational complexity and reconstruction granularity in certain implementations also highlighted opportunities for future research.

Acknowledgments

This project has been partially funded by the Horizon EU project TRUSTroke in the call HORIZON-HLTH-2022-STAYHLTH-01-two-stage under GA No. 101080564. and by the Italian Ministry of University and Research (MUR) under the PRIN 2022 PNRR framework (EU Contribution – Next Generation EU – M. 4,C. 2, I. 1.1), SHIELDED project, ID P2022ZWS82, CUP D53D23016240001.

Generative AI tools such as Grammarly and ChatGPT (GPT-4o) were utilized solely for proofreading and grammar refinement in the preparation of this manuscript. The authors retain full responsibility for the content presented in the final version.

References

[1] Mustafa Abdul Salam, Khaled M. Fouad, Doaa L. Elbably, and Salah M. Elsayed. 2024. Federated learning model for credit card fraud detection with data balancing techniques. *Neural Computing and Applications* 36, 11 (April 2024), 6231–6256. https://doi.org/10.1007/s00521-023-09410-2

[2] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2018. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Comput. Surv.* 51, 4 (July 2018), 79:1–79:35. https://doi.org/10.1145/3214303

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How To Backdoor Federated Learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 108)*, Silvia Chiappa and Roberto Calandra (Eds.). PMLR, 2938–2948. https://proceedings.mlr.press/v108/bagdasaryan20a.html

[4] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. 2022. Flower: A Friendly Federated Learning Research Framework. https://doi.org/10.48550/arXiv.2007.14390 arXiv:2007.14390 [cs].

[5] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2016. Practical Secure Aggregation for Federated Learning on User-Held Data. https://doi.org/10.48550/arXiv.1611.04482 arXiv:1611.04482 [cs].

[6] Bernardo Camajori Tedeschini, Stefano Savazzi, Roman Stoklasa, Luca Barbieri, Ioannis Stathopoulos, Monica Nicoli, and Luigi Serio. 2022. Decentralized Federated Learning for Healthcare Networks: A Case Study on Tumor Segmentation. *IEEE Access* 10 (2022), 8693–8708. https://doi.org/10.1109/ACCESS.2022.3141913

[7] Centers for Medicare & Medicaid Services. 1996. The Health Insurance Portability and Accountability Act of 1996 (HIPAA). Published: Online at https://www.hhs.gov/hipaa.

[8] Sylvain Chatel, Apostolos Pyrgelis, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. 2021. SoK: Privacy-Preserving Collaborative Tree-based Model Learning. *Proceedings on Privacy Enhancing Technologies* 3 (2021), 182–203. https://doi.org/10.2478/popets-2021-0043

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16 (June 2002), 321–357. https://doi.org/10.1613/jair.953

[10] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[11] Tianqi Chen, Sameer Singh, Ben Taskar, and Carlos Guestrin. 2015. Efficient Second-Order Gradient Boosting for Conditional Random Fields. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 38)*, Guy Lebanon and S. V. N. Vishwanathan (Eds.). PMLR, San Diego, California, USA, 147–155. https://proceedings.mlr.press/v38/chen15b.html

[12] Xiaolin Chen, Daoguang Zan, Wei Li, Bei Guan, and Yongji Wang. 2024. FIA-TE: Feature Inference Attack on Decision Tree Ensembles in Vertical Federated Learning. In *2024 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6. https://doi.org/10.1109/ICME57554.2024.10687832

[13] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. 2021. SecureBoost: A Lossless Federated Learning Framework. *IEEE Intelligent Systems* 36, 6 (2021), 87–98. https://doi.org/10.1109/MIS.2021.3082561

[14] Dilip Kumar Choubey, Sanchita Paul, Santosh Kumar, and Shankar Kumar. 2017. Classification of Pima indian diabetes dataset using naive bayes with genetic algorithm as an attribute selection. In *Communication and computing systems: proceedings of the international conference on communication and computing system (ICCCS 2016)*. 451–455.

[15] Kelong Cong, Debajyoti Das, Jeongeun Park, and Hilder V.L. Pereira. 2022. SortingHat: Efficient Private Decision Tree Evaluation via Homomorphic Encryption and Transciphering. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 563–577. https://doi.org/10.1145/3548606.3560702

[16] Thien-Phuc Doan, Bong Jun Choi, Kihun Hong, Jungsoo Park, and Souhwan Jung. 2023. Random Forest in Federated Learning Setting. In *Advances in Computer Science and Ubiquitous Computing*, Ji Su Park, Laurence T. Yang, Yi Pan, and Jong Hyuk Park (Eds.). Vol. 1028. Springer Nature Singapore, Singapore, 1–9. https://doi.org/10.1007/978-981-99-1252-0_1 Series Title: Lecture Notes in Electrical Engineering.

[17] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12.

[18] David Enthoven and Zaid Al-Ars. 2020. An Overview of Federated Deep Learning Privacy Attacks and Defensive Strategies. http://arxiv.org/abs/2004.04676 arXiv:2004.04676 [cs, stat].

[19] Yann Fraboni, Richard Vidal, and Marco Lorenzi. 2021. Free-rider Attacks on Model Aggregation in Federated Learning. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 130)*, Arindam Banerjee and Kenji Fukumizu (Eds.). PMLR, 1846–1854. https://proceedings.mlr.press/v130/fraboni21a.html

[20] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* 29, 5 (Oct. 2001). https://doi.org/10.1214/aos/1013203451

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[22] Gurobi Optimization, LLC. 2024. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[23] Hafsa Habehh and Suril Gohel. 2021. Machine Learning in Healthcare. *Current Genomics* 22, 4 (Dec. 2021), 291–300. https://doi.org/10.2174/1389202922666210705124359

[24] Ahmad Hassan. 2023. Stroke Prediction Dataset. https://doi.org/10.21227/mxfb-sc71

[25] Anne-Christin Hauschild, Marta Lemanczyk, Julian Matschinske, Tobias Frisch, Olga Zolotareva, Andreas Holzinger, Jan Baumbach, and Dominik Heider. 2022. Federated Random Forests can improve local performance of predictive models for various healthcare applications. *Bioinformatics* 38, 8 (April 2022), 2278–2286. https://doi.org/10.1093/bioinformatics/btac065

[26] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 603–618. https://doi.org/10.1145/3133956.3134012

[27] Zhanglong Ji, Zachary C. Lipton, and Charles Elkan. 2014. Differential Privacy and Machine Learning: a Survey and Review. https://doi.org/10.48550/arXiv.1412.7584 arXiv:1412.7584 [cs].

[28] H. W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1-2 (1955), 83–97. https://doi.org/10.1002/nav.3800020109

[29] K Naveen Kumar, C Krishna Mohan, and Linga Reddy Cenkeramaddi. 2023. The Impact of Adversarial Attacks on Federated Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023), 1–20. https://doi.org/10.1109/TPAMI.2023.3322785

[30] Andrew Law, Chester Leung, Rishabh Poddar, Raluca Ada Popa, Chenyu Shi, Octavian Sima, Chaofan Yu, Xingmeng Zhang, and Wenting Zheng. 2020. Secure Collaborative Training and Inference for XGBoost. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice* (Virtual Event, USA) *(PPMLP'20)*. Association for Computing Machinery, New York, NY, USA, 21–26. https://doi.org/10.1145/3411501.3419420

[31] Mengqian Li, Youliang Tian, Junpeng Zhang, Dandan Fan, and Dongmei Zhao. 2021. The Trade-off Between Privacy and Utility in Local Differential Privacy. In *2021 International Conference on Networking and Network Applications (NaNA)*. 373–378. https://doi.org/10.1109/NaNA53684.2021.00071

[32] Qinbin Li, Zeyi Wen, and Bingsheng He. 2020. Practical Federated Gradient Boosting Decision Trees. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (Apr. 2020), 4642–4649. https://doi.org/10.1609/aaai.v34i04.5895

[33] Qinbin Li, WU ZHAOMIN, Yanzheng Cai, yuxuan han, Ching Man Yung, Tianyuan Fu, and Bingsheng He. 2023. FedTree: A Federated Learning System For Trees. In *Proceedings of Machine Learning and Systems*, D. Song, M. Carbin, and T. Chen (Eds.), Vol. 5. Curan, 89–103. https://proceedings.mlsys.org/paper_files/paper/2023/file/3430e7055936cb8e26451ed49fce84a6-Paper-mlsys2023.pdf

[34] Yehuda Lindell and Benny Pinkas. 2009. Secure Multiparty Computation for Privacy-Preserving Data Mining. *Journal of Privacy and Confidentiality* 1, 1 (Apr. 2009). https://doi.org/10.29012/jpc.v1i1.566

[35] William Lindskog-Münzing and Christian Prehofer. 2024. treeXnets: Comparing Federated Tree-BasedModels and Neural Networks on Tabular Data. https://doi.org/10.21203/rs.3.rs-4499006/v1

[36] Pengrui Liu, Xiangrui Xu, and Wei Wang. 2022. Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. *Cybersecurity* 5, 1 (Dec. 2022), 4. https://doi.org/10.1186/s42400-021-00105-6

[37] Yang Liu, Zhuo Ma, Ximeng Liu, Siqi Ma, Surya Nepal, Robert. H Deng, and Kui Ren. 2020. Boosting Privately: Federated Extreme Gradient Boosting for Mobile Crowdsensing. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. 1–11. https://doi.org/10.1109/ICDCS47774.2020.00017

[38] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2021. Feature Inference Attack on Model Predictions in Vertical Federated Learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, Chania, Greece, 181–192. https://doi.org/10.1109/ICDE51399.2021.00023

[39] Chenyang Ma, Xinchi Qiu, Daniel Beutel, and Nicholas Lane. 2023. Gradient-less Federated Gradient Boosting Tree with Learnable Learning Rates. In *Proceedings of the 3rd Workshop on Machine Learning and Systems* (Rome, Italy) *(EuroMLSys '23)*. Association for Computing Machinery, New York, NY, USA, 56–63. https://doi.org/10.1145/3578356.3592579

[40] Samuel Maddock, Graham Cormode, Tianhao Wang, Carsten Maple, and Somesh Jha. 2022. Federated Boosted Decision Trees with Differential Privacy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Los Angeles CA USA, 2249–2263. https://doi.org/10.1145/3548606.3560687

[41] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

[42] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting Unintended Feature Leakage in Collaborative Learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 691–706. https://doi.org/10.1109/SP.2019.00029

[43] Travis B. Murdoch and Allan S. Detsky. 2013. The Inevitable Application of Big Data to Health Care. *JAMA* 309, 13 (April 2013), 1351. https://doi.org/10.1001/jama.2013.393

[44] Hanchi Ren, Jingjing Deng, and Xianghua Xie. 2022. GRNN: Generative Regression Neural Network—A Data Leakage Attack for Federated Learning. *ACM Trans. Intell. Syst. Technol.* 13, 4, Article 65 (May 2022), 24 pages. https://doi.org/10.1145/3510032

[45] Ryan Rifkin and Aldebaro Klautau. 2004. In Defense of One-Vs-All Classification. *J. Mach. Learn. Res.* 5 (Dec. 2004), 101–141.

[46] Maria Rigaki and Sebastian Garcia. 2023. A Survey of Privacy Attacks in Machine Learning. *ACM Comput. Surv.* 56, 4, Article 101 (Nov. 2023), 34 pages. https://doi.org/10.1145/3624010

[47] Lior Rokach and Oded Maimon. 2005. Decision Trees. In *Data Mining and Knowledge Discovery Handbook*, Oded Maimon and Lior Rokach (Eds.). Springer-Verlag, New York, 165–192. https://doi.org/10.1007/0-387-25465-X_9

[48] Holger R. Roth, Yan Cheng, Yuhong Wen, Isaac Yang, Ziyue Xu, Yuan-Ting Hsieh, Kristopher Kersten, Ahmed Harouni, Can Zhao, Kevin Lu, Zhihong Zhang, Wenqi Li, Andriy Myronenko, Dong Yang, Sean Yang, Nicola Rieke, Abood Quraini, Chester Chen, Daguang Xu, Nic Ma, Prerna Dogra, Mona Flores, and Andrew Feng. 2022. NVIDIA FLARE: Federated Learning from Simulation to Real-World. (2022). https://doi.org/10.48550/arXiv.2210.13291 arXiv:2210.13291 [cs].

[49] Diogo Reis Santos, Albert Sund Aillet, Antonio Boiano, Usevalad Milasheuski, Lorenzo Giusti, Marco Di Gennaro, Sanaz Kianoush, Luca Barbieri, Monica Nicoli, Michele Carminati, Alessandro E. C. Redondi, Stefano Savazzi, and Luigi Serio. 2024. A Federated Learning Platform as a Service for Advancing Stroke Management in European Clinical Centers. In *2024 IEEE International Conference on E-health Networking, Application & Services (HealthCom)*. 1–7. https://doi.org/10.1109/HealthCom60970.2024.10880750

[50] S. Sinharay. 2010. Continuous Probability Distributions. In *International Encyclopedia of Education*. Elsevier, 98–102. https://doi.org/10.1016/B978-0-08-044894-7.01720-6

[51] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. 2019. Can You Really Backdoor Federated Learning? http://arxiv.org/abs/1911.07963 arXiv:1911.07963 [cs, stat].

[52] Hideaki Takahashi, Jingjing Liu, and Yang Liu. 2023. Eliminating Label Leakage in Tree-Based Vertical Federated Learning. https://doi.org/10.48550/arXiv.2307.10318 arXiv:2307.10318 [cs].

[53] Zhihua Tian, Rui Zhang, Xiaoyang Hou, Lingjuan Lyu, Tianyi Zhang, Jian Liu, and Kui Ren. 2024. FederBoost: Private Federated Learning for GBDT. *IEEE Transactions on Dependable and Secure Computing* 21, 3 (2024), 1274–1285. https://doi.org/10.1109/TDSC.2023.3276365

[54] Mark Vero, Mislav Balunović, Dimitar I. Dimitrov, and Martin Vechev. 2023. TabLeak: tabular data leakage in federated learning. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 1460, 33 pages.

[55] Paul Voigt and Axel von dem Bussche. 2017. *The EU General Data Protection Regulation (GDPR): A Practical Guide* (1st ed.). Springer Publishing Company, Incorporated.

[56] Zijun Wang and Keke Gai. 2024. Decision Tree-Based Federated Learning: A Survey. *Blockchains* 2, 1 (March 2024), 40–60. https://doi.org/10.3390/blockchains2010003

[57] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications* (Paris, France). IEEE Press, 2512–2520. https://doi.org/10.1109/INFOCOM.2019.8737416

[58] Han Wu, Zilong Zhao, Lydia Y. Chen, and Aad Van Moorsel. 2022. Federated Learning for Tabular Data: Exploring Potential Risk to Privacy. In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, Charlotte, NC, USA, 193–204. https://doi.org/10.1109/ISSRE55969.2022.00028

[59] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. 2020. Privacy preserving vertical federated learning for tree-based models. *Proceedings of the VLDB Endowment* 13, 12 (Aug. 2020), 2090–2103. https://doi.org/10.14778/3407790.3407811

[60] Jin Xu, Chi Hong, Jiyue Huang, Lydia Y. Chen, and Jeremie Decouchant. 2022. AGIC: Approximate Gradient Inversion Attack on Federated Learning . In *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*. IEEE Computer Society, Los Alamitos, CA, USA, 12–22. https://doi.org/10.1109/SRDS55811.2022.00012

[61] Fuki Yamamoto, Seiichi Ozawa, and Lihua Wang. 2022. eFL-Boost: Efficient Federated Learning for Gradient Boosting Decision Trees. *IEEE Access* 10 (2022), 43954–43963. https://doi.org/10.1109/ACCESS.2022.3169502

[62] Fuki Yamamoto, Lihua Wang, and Seiichi Ozawa. 2020. New Approaches to Federated XGBoost Learning for Privacy-Preserving Data Analysis. In *Neural Information Processing*, Haiqin Yang, Kitsuchart Pasupa, Andrew Chi-Sing Leung, James T. Kwok, Jonathan H. Chan, and Irwin King (Eds.). Springer International Publishing, Cham, 558–569. https://doi.org/10.1007/978-3-030-63833-7_47

[63] Haomiao Yang, Mengyu Ge, Kunlan Xiang, and Jingwei Li. 2023. Using Highly Compressed Gradients in Federated Learning for Data Reconstruction Attacks. *IEEE Transactions on Information Forensics and Security* 18 (2023), 818–830. https://doi.org/10.1109/TIFS.2022.3227761

[64] Mengwei Yang, Linqi Song, Jie Xu, Congduan Li, and Guozhen Tan. 2019. The Tradeoff Between Privacy and Accuracy in Anomaly Detection Using Federated XGBoost. https://doi.org/10.48550/arXiv.1907.07157 arXiv:1907.07157 [cs].

[65] Boyang Zhang, Zheng Li, Ziqing Yang, Xinlei He, Michael Backes, Mario Fritz, and Yang Zhang. 2024. SecurityNet: Assessing Machine Learning Vulnerabilities on Public Models. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA, 3873–3890. https://www.usenix.org/conference/usenixsecurity24/presentation/zhang-boyang

[66] Chuan Zhang, Haotian Liang, Youqi Li, Tong Wu, Liehuang Zhu, and Weiting Zhang. 2023. Stealing Secrecy from Outside: A Novel Gradient Inversion Attack in Federated Learning. In *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, Nanjing, China, 282–288. https://doi.org/10.1109/ICPADS56603.2022.00044

[67] Joshua C. Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. 2024. Loki: Large-scale Data Reconstruction Attack against Federated Learning through Model Manipulation. In *2024 IEEE Symposium on Security and Privacy (SP)*. 1287–1305. https://doi.org/10.1109/SP54263.2024.00030 ISSN: 2375-1207.

[68] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf

A Ethical Considerations

This research was conducted with a clear commitment to ethical responsibility, carefully considering the impact on all relevant stakeholders. The decisions made throughout the study balanced transparency, security, and long-term benefits. We are aware that the publication of attacks may raise ethical concerns, as it exposes techniques that could be exploited by malicious actors. However, failing to address these risks would leave users unaware of real-world threats. By presenting our research, we highlight the privacy risks of tree-based Federated Learning while also providing mitigation guidelines, including insights into the limitations of differential

Table 4: Reconstruction Accuracy (RA) after the First-Tree Probing phase (F-TP) [%], RA after the complete attack[%], F1-score, and Area Under the ROC Curve (AUC-ROC) on the test set for the binary classification task on the Stroke Prediction Dataset. Each result corresponds to the tree depth d used during training.

	Local Reconstruction												Global Reconstruction							
Depth	FedXGBllr				Flower XGBoost Bagging				Flower XGBoost Cyclic				NVFlare				FedTree			
	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC
3	74.54	84.28	0.215	0.635	72.01	76.00	0.213	0.755	69.74	73.05	0.237	0.748	74.28	85.93	0.165	0.767	74.53	84.91	0.207	0.766
4	75.93	85.90	0.218	0.628	74.78	80.33	0.215	0.750	72.48	79.90	0.157	0.738	75.90	86.86	0.203	0.774	75.79	86.47	0.200	0.764
5	76.41	87.47	0.183	0.620	75.89	77.02	0.185	0.750	75.22	80.57	0.158	0.731	76.35	85.90	0.203	0.767	77.27	85.99	0.209	0.766
6	77.54	85.85	0.157	0.617	76.13	80.97	0.176	0.746	76.21	79.51	0.162	0.760	76.85	83.66	0.164	0.775	77.29	87.63	0.110	0.767
7	78.40	86.53	0.105	0.608	76.99	82.24	0.138	0.731	76.55	83.33	0.176	0.734	77.06	82.07	0.103	0.744	78.59	85.85	0.130	0.768
8	79.19	85.64	0.097	0.614	78.70	82.63	0.160	0.751	77.74	82.98	0.158	0.709	76.80	82.80	0.128	0.761	79.03	87.56	0.108	0.769

Table 5: Reconstruction Accuracy (RA) (top 50% important columns) after the First-Tree Probing phase (F-TP) [%], RA (top 50% important columns) after the complete attack [%] for the binary classification task on the Stroke Prediction Dataset. Each result corresponds to the tree depth d used during training.

	Local Reconstruction						Global Reconstruction			
Depth	FedXGBllr		Flower XGBoost Bagging		Flower XGBoost Cyclic		NVFlare		FedTree	
	RA (F-TP)	RA	RA (F-TP)	RA	RA (F-TP)	RA	RA (F-TP)	RA	RA (F-TP)	RA
3	76.50	88.89	75.95	85.74	76.02	83.55	75.90	94.10	77.15	92.23
4	79.10	91.34	78.65	83.83	76.63	87.13	78.29	91.55	78.71	92.33
5	78.52	95.63	80.42	84.00	77.79	88.40	79.11	93.05	80.42	93.84
6	79.59	91.95	78.82	86.92	77.46	86.66	80.20	89.42	81.99	94.71
7	82.64	95.15	81.25	87.59	78.98	89.63	78.80	86.79	83.38	92.49
8	82.72	91.34	82.95	91.05	79.45	89.38	75.70	86.57	83.74	94.80

Table 6: Reconstruction Accuracy (RA) after the First-Tree Probing phase (F-TP) [%], RA after the complete attack [%], F1-score, and Area Under the ROC Curve (AUC-ROC) on the test set for the binary classification task on the Pima Indians Diabetes Dataset. Each result corresponds to the tree depth d used during training.

	Local Reconstruction												Global Reconstruction							
Depth	FedXGBllr				Flower XGBoost Bagging				Flower XGBoost Cyclic				NVFlare				FedTree			
	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC
3	63.82	68.20	0.587	0.816	59.85	57.56	0.638	0.769	57.07	54.45	0.573	0.753	60.57	65.81	0.643	0.817	63.08	71.74	0.586	0.810
4	60.57	66.36	0.603	0.806	62.34	58.02	0.587	0.743	64.20	59.74	0.506	0.669	60.20	66.01	0.676	0.834	63.21	72.74	0.596	0.826
5	63.00	66.61	0.572	0.810	61.55	62.05	0.577	0.716	61.93	59.82	0.490	0.669	61.97	67.38	0.639	0.821	65.98	72.51	0.606	0.798
6	64.70	67.42	0.531	0.806	62.71	64.81	0.506	0.689	62.83	58.29	0.591	0.755	63.79	67.87	0.636	0.831	66.16	73.27	0.604	0.805
7	64.52	65.94	0.607	0.809	63.32	64.03	0.537	0.703	63.85	60.35	0.549	0.719	63.26	64.41	0.627	0.832	67.35	73.03	0.604	0.806
8	64.24	67.14	0.582	0.814	63.32	56.61	0.585	0.718	64.38	62.70	0.529	0.679	63.84	65.94	0.660	0.825	67.46	73.37	0.610	0.815

Table 7: Reconstruction Accuracy (RA) (top 50% important columns) after the First-Tree Probing phase (F-TP) [%], RA (top 50% important columns) after the complete attack [%] for the binary classification task on the Pima Indians Diabetes Dataset. Each result corresponds to the tree depth d used during training.

	Local Reconstruction						Global Reconstruction			
Depth	FedXGBllr		Flower XGBoost Bagging		Flower XGBoost Cyclic		NVFlare		FedTree	
	RA (F-TP)	RA	RA (F-TP)	RA	RA (F-TP)	RA	RA (F-TP)	RA	RA (F-TP)	RA
3	78.39	82.57	78.26	72.67	71.55	71.69	75.62	84.20	79.67	87.87
4	77.44	79.21	79.37	74.91	78.56	75.86	78.04	83.82	80.67	86.32
5	80.39	80.74	79.68	77.97	80.36	75.44	79.10	83.03	83.71	87.57
6	79.68	79.98	79.80	76.62	79.09	74.04	80.62	84.07	82.95	88.17
7	79.74	79.15	80.33	77.09	80.33	75.97	81.13	83.58	84.07	87.02
8	79.27	80.27	81.18	72.26	80.39	79.69	83.14	84.45	84.80	88.57

Table 8: FedTree with Differential Privacy (DP) on different values of privacy budget ϵ (Both histogram-level and total ϵ are reported). Reconstruction Accuracy (RA) after the First-Tree Probing phase (F-TP) [%], RA after the complete attack [%], F1-score, and Area Under the ROC Curve (AUC-ROC) on the test set for the binary classification task on the Pima Indians Diabetes Dataset. Each result corresponds to the tree depth d used during training.

Depth	No Defense				$\epsilon_{\text{histogram}} = 1 \mid \epsilon_{\text{total}} = 200$				$\epsilon_{\text{histogram}} = 0.25 \mid \epsilon_{\text{total}} = 50$				$\epsilon_{\text{histogram}} = 0.125 \mid \epsilon_{\text{total}} = 25$			
	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC	RA (F-TP)	RA	F1	AUC
3	63.08	71.74	0.586	0.810	54.32	67.48	0.500	0.811	53.88	57.43	0.471	0.645	56.47	48.31	0.495	0.694
4	63.21	72.74	0.596	0.826	58.84	69.41	0.589	0.819	57.02	61.89	0.415	0.614	60.75	50.75	0.481	0.641
5	65.98	72.51	0.606	0.798	59.92	71.14	0.568	0.798	60.37	58.53	0.547	0.708	60.54	55.94	0.472	0.599
6	66.16	73.27	0.604	0.805	62.39	71.38	0.531	0.739	59.90	61.64	0.468	0.681	61.56	52.82	0.386	0.589
7	67.35	73.03	0.604	0.806	62.43	71.29	0.529	0.769	63.94	64.12	0.496	0.626	61.55	58.18	0.417	0.555
8	67.46	73.37	0.610	0.815	65.99	72.39	0.500	0.751	64.69	65.93	0.487	0.624	60.70	60.55	0.435	0.550

Table 9: FedTree with Differential Privacy (DP) on different values of privacy budget ϵ (Both histogram-level and total ϵ are reported). Reconstruction Accuracy (RA) (top 50% important columns) after the First-Tree Probing phase (F-TP) [%], RA (top 50% important columns) after the complete attack [%] for the binary classification task on the Pima Indians Diabetes Dataset. Each result corresponds to the tree depth d used during training.

Depth	No Defense		$\epsilon_{\text{histogram}} = 1 \mid \epsilon_{\text{total}} = 200$		$\epsilon_{\text{histogram}} = 0.25 \mid \epsilon_{\text{total}} = 50$		$\epsilon_{\text{histogram}} = 0.125 \mid \epsilon_{\text{total}} = 25$	
	RA (F-TP)	RA	RA (F-TP)	RA	RA (F-TP)	RA	RA (F-TP)	RA
3	79.67	87.87	72.01	84.77	59.69	72.37	68.05	60.21
4	80.67	86.32	71.55	84.12	69.73	75.95	74.00	65.93
5	83.71	87.57	74.62	86.24	74.59	70.30	74.02	70.30
6	82.95	88.17	78.20	87.02	75.43	75.05	77.74	63.22
7	84.07	87.02	77.93	85.97	78.77	78.75	76.17	70.98
8	84.80	88.57	81.38	86.92	81.35	81.76	75.08	75.24

privacy and other defenses, which may not be as effective as users assume. This approach ensures that users can take informed security measures. Responsible disclosure of threats, paired with the provision of practical countermeasures, helps prevent adversaries from gaining an asymmetric advantage. Our work follows this principle by equipping the community with both an understanding of potential risks and the means to mitigate them.

B Background

In this section, we complete the background by providing details about Differential Privacy (DP).

B.1 Differential Privacy

Differential Privacy (DP) [17, 27] is a privacy definition that ensures that the output of a computation does not reveal too much information about any individual in the dataset. The idea is to add noise to the output of the computation in such a way that the privacy of the individuals is preserved. Formally, a randomized algorithm $\mathcal{M}$ satisfies ϵ-differential privacy if for all datasets D and D' that differ in one element, and for all subsets of the output space S:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in S]. \tag{16}$$

The parameter ϵ, also called *privacy budget*, is a measure of the privacy loss; the smaller the value of ϵ, the more privacy is preserved. The Laplace mechanism is a simple way to achieve DP by adding Laplace noise to the output of the computation. The Laplace mechanism is defined as:

$$\mathcal{M}(D) = f(D) + \text{Lap}(\frac{\Delta f}{\epsilon}), \tag{17}$$

where $f(D)$ is the output of the computation on the dataset D, Δf is the sensitivity of the function f, and $\text{Lap}(\lambda)$ is the Laplace distribution with scale λ.

B.1.1 FedTree ϵ-DP implementation. The FedTree [33] protocol implies the sharing of histograms at each round. Therefore, it implements ϵ-DP at the level of histogram sharing. Unlike FL protocols based on Artificial Neural Networks, where the gradients or model weights are the objects of protection, in FedTree, the sensitive information resides in the *histograms*, which contain the aggregated gradient and Hessian statistics used to construct tree nodes.

To satisfy ϵ-DP for each shared histogram, FedTree follows the procedure we explain below.

Gradient Clipping. First-order gradients and the second-order ones are clipped with a threshold of R and $2R$ respectively, to ensure bounded sensitivity.

Laplace Noise Addition. Laplace noise is added to each element of the histogram. Specifically, the noise is drawn from the Laplace distribution with mean 0 and scale $\frac{2R}{\epsilon}$, i.e., $\text{Lap}(0, \frac{2R}{\epsilon})$. This guarantees that each shared histogram satisfies ϵ-DP.

Histogram-Level Privacy. DP is applied *per histogram*, meaning that the privacy budget $\epsilon_{\text{histogram}}$ governs the noise added to each update (i.e., histograms), rather than to the entire model.

Total Privacy Budget. In their implementation[2], FedTree allows setting an aggregated privacy budget ϵ_{total}, which represents the upper bound on the cumulative privacy loss across all trees. As a result, this budget depends on the number of trees. Specifically,

[2] https://github.com/Xtra-Computing/FedTree

given a value for ϵ_{total}, the privacy budget allocated to each leaf node histogram, $\epsilon_{\text{histogram}}$, is computed using the following formula:

$$\epsilon_{\text{histogram}} = \frac{\epsilon_{\text{total}}}{2T} \tag{18}$$

where T denotes the number of trees in the model.

C Detailed Results

In this section, we provide the raw results of experiments in Sections 6.2 and 6.3. Specifically, the results regarding Section 6.2 for the Stroke Dataset are presented in Tables 4 and 5, while the results for the Diabetes Dataset are shown in Tables 6 and 7. Finally, the results regarding Section 6.3 are presented in Tables 8 and 9.

D Attack Computational Complexity

In this section, we analyze the computational complexity of our proposed reconstruction attack. As shown in Section 5, our attack consists of two phases: *First-Tree Probing* and *Feature Range Inference*. In the following, we theoretically analyze the complexity of each phase.

First-Tree Probing. In this phase, intermediate values are computed for each leaf, requiring $O(L)$ operations, where L denotes the number of leaves in the DT and d its depth. Additionally, for each leaf, the DT is traversed to aggregate the corresponding feature range information. Since each traversal requires $O(\log d)$ time, processing all L leaves results in an additional cost of $O(L \cdot \log d)$. Therefore, the overall computational complexity of the *First-Tree Probing* phase is $O(L \cdot \log d)$.

Feature Range Inference. This phase involves solving a series of Mixed-Integer Linear Programming (MILP) problems for binary decision-making. In general, MILP is NP-hard and has a worst-case computational complexity that is exponential in the number of binary decision variables. In our formulation, each variable x_{ij} indicates whether sample i is assigned to leaf j. If every sample between the n samples in the dataset could be assigned to any of the m leaves, the search space would contain $O(n \cdot m)$ binary variables, leading to exponential complexity in the worst case. However, our model introduces constraints of the form $x_{ij} = 0$ for all $j \notin L_i$, where $L_i \subseteq J$ denotes the subset of leaves that sample i can reach, based on inferred feature ranges. As the attack progresses through successive trees, these feature ranges become increasingly constrained, reducing the size of L_i for each sample. This prunes the effective search space, significantly mitigating the theoretical worst-case complexity in practical scenarios. To further control optimization time, during our experimental evaluation, we impose a 10-minute time constraint on the MILP solver for each tree. This ensures that the optimization phase remains computationally feasible, even when the worst-case complexity is prohibitive, while having a minimal impact on the overall reconstruction quality.

E Extension to Multiclass Classification

In Section 5, we provided a formalization of our attack against federated gradient boosting binary classifiers. In this section, we extend the theoretical formalization to multiclass classification by highlighting the differences with the binary case.

Gradient Boosting Decision Trees (GBDT) [20] and XGBoost [10] approach the multiclass classification task using a One-vs-Rest (OvR) strategy [45]. In this approach, each boosting iteration trains K trees—one for each of the K classes—where each tree is trained to distinguish one class from all the others. Unlike the binary classification case, which typically uses the log loss function, the multiclass setting employs a softmax loss function. This allows each class-specific tree to be optimized by considering the predictions from all K class-specific trees from the previous boosting iteration.

Regarding federated implementations of multiclass algorithms, their mechanisms are equivalent to those used for binary classification. Therefore, the information available on the client side remains the same as discussed in the binary case.

We now describe the two phases of TimberStrike (*First-Tree Probing* and *Feature Range Inference*) adapted to the multiclass setting.

E.1 First-Tree Probing

As in the binary classification case, the controlled client targets a specific client and analyzes the first trees trained by that client. In the multiclass scenario, this corresponds to analyzing the first K trees, one for each of the K classes. Each of these trees distinguishes between class 1 (the class c for which the tree is trained) and class 0 (all other classes).

The base score in the multiclass case differs from that in the binary case. Indeed, while in binary classification it is directly interpreted as a probability, in the multiclass setting the base score acts as a logit and is the same for all K classes. Therefore, the initial probability for each class is:

$$p_i^{(c)} = \frac{1}{K}, \quad \forall c \in \{1, \ldots, K\}.$$

As discussed in Section 5.1, the *First-Tree Probing* phase consists of four steps, which we now describe in the multiclass context.

Inferring the Number of Samples per Leaf. After clients receive the aggregated trees from the PS, the adversary can infer the number of samples assigned to each leaf by analyzing the first K trees trained by the victim client. In the multiclass setting, the loss function is the softmax cross-entropy, and we use its derivatives [11, 21] to compute the per-sample gradient and Hessian statistics.

For a sample i and class c, the gradient and Hessian value are:

$$g_i^{(c)} = p_i^{(c)} - \mathbb{1}[y_i = c], \quad h_i^{(c)} = 2 \cdot p_i^{(c)} \cdot (1 - p_i^{(c)}), \tag{19}$$

where $p_i^{(c)}$ is the softmax probability for class c. The total Hessian $H_j^{(c)}$ in a leaf j of the first tree for class c can then be expressed as:

$$H_j^{(c)} = \sum_{i=1}^{N_j} h_{ij}^{(c)} = N_j \cdot 2 \cdot p_i^{(c)} \cdot (1 - p_i^{(c)}), \tag{20}$$

and solving for N_j (the number of samples in the leaf) we get:

$$N_j = \frac{H_j^{(c)}}{2 \cdot p_i^{(c)} \cdot (1 - p_i^{(c)})}. \tag{21}$$

Inferring the Label Distribution in the Leaves. After inferring the number of samples N_j in a leaf j, we recover the label

distribution using the gradient for each leaf j and class c:

$$G_j^{(c)} = -\frac{leaf\_value_j^{(c)}}{\eta} \cdot (H_j^{(c)} + \lambda), \tag{22}$$

where η is the learning rate and λ is the regularization parameter.

Using the per-class aggregated gradient expression, we compute the number of samples $N_j^{(c)}$ for class c in the leaf j as:

$$N_j^{(c)} = N_j \cdot p_i^{(c)} - G_j^{(c)}. \tag{23}$$

Computing this for all K classes (i.e., K trees), we obtain the label distribution in each leaf, where 1 represents the class c for which the tree has been trained, while 0 represents each other class.

Dataset Initialization. At this point, the adversary knows the number of samples per leaf and their class distribution. As in the binary case, they can initialize a dataset by placing the appropriate number of samples with the inferred class labels into each leaf. Tree paths are again used to constrain feature ranges, and samples are initialized uniformly within those ranges.

However, in the multiclass setting, the "path traversal" used to initialize feature ranges is not performed for every sample in the first K trees. Instead, for each client-specific tree corresponding to class c, only the samples with inferred label $y_i = c$ are initialized and assigned feature constraints.

Compute per-sample Statistics. Finally, for each inferred sample and class, the adversary computes the probability score $p_i^{(c)}$ with the softmax function:

$$p_i^{(c)} = \text{softmax}\left(base\_score + \sum_{t=1}^{T} leaf\_value_i^{(c,t)}\right), \tag{24}$$

where $\text{leaf\_value}_i^{(c,t)}$ is the value of the leaf reached by sample i in the t-th tree for class c, and T is the number of boosting rounds.

The gradients and Hessian values are then computed using Equations (19), and are used in the second phase of the attack.

E.2 Feature Range Inference

In the multiclass setting, the second phase of TimberStrike proceeds similarly to the binary case. Since each boosting iteration generates K class-specific trees (one per class), the adversary solves one optimization problem for each of these trees. However, thanks to the OvR strategy, each tree can be treated as a binary classifier that distinguishes class c from the rest. Therefore, the same Mixed-Integer Linear Programming (MILP) formalization described for the binary case can be applied directly, provided that the appropriate values of $p_i^{(c)}$, $g_i^{(c)}$, and $h_i^{(c)}$ are used for each client-specific tree corresponding to class c. In addition, since the client-specific trees are built in parallel during training, the per-sample statistics can be updated only after both the optimization problem and the feature range update have been completed for each of the K trees in the same boosting iteration.

As in the binary case, this iterative refinement improves the accuracy of the reconstructed dataset by refining the feature ranges based on the per-tree leaf assignments.