Project Group Work in Cybersecurity M

PYOD: Pwn Your Own Device

Riccardo Bovinelli, Luca Cimino, Lorenzo Riccardi,
Tommaso Sgreccia

December 2024

Contents

Abstract

Federated Learning is a rapidly growing Machine Learning technique that focuses on distributed learning running on heterogeneous devices. Due to its distributed nature, it is subject to a variety of vulnerabilities. After a review of the main threat vectors and vulnerabilities of Federated Learning, in this work we present an analysis of the state-of-the-art of the attacks to Federated Learning, from those based on data to those that exploit the intrinsic vulnerabilities of the federation. Furthermore, we present an overview on the main defense techniques used to counteract these malicious actions. We also provide an explanation and performance statistics of three different implemented attacks: label flipping, an inference attack and a model poisoning attack, both based on Generative Adversarial Networks. Finally, we provide some mitigation approaches for these specific attacks, based on the list of state-of-the-art defenses that we discussed earlier.

1 Introduction

1.1 Centralized Machine Learning

Machine Learning (ML) is the process of developing and creating algorithms and models that can learn through training on a dataset, and then applying their knowledge to a set of unseen data. This is achieved through the application of various operations on the data. A variety of models may be employed, including neural networks and decision trees, among others. The data used during the training phase are not typically generated on the machine where the model is trained; rather, they originate from numerous devices (such as IoT sensors).
In a classic Machine Learning system, once created, data is processed in a central server where the ML model is located, through a process called *training*. Once trained new data can be submitted to the model which will perform *predictions* on them.

1.1.1 Limitations of the centralized approach

The classical centralized approach can be applied to many different scenarios, such as Web Analysis, but it may not be always the appropriate solution. In distributed systems, many devices like IoT sensors and smartphones, often handle private and sensitive data that should remain on the device for privacy reasons. Centralizing these data would introduce the risk of exposing it to third-party entities, leading to significant privacy concerns. Additionally, compliance with privacy governance and regulations over different parts of the world, limit the the transfer and sharing of such data across different locations.
Furthermore, for cases where the data is not available on a centralized server, or other cases where the data available on a single server is not enough to train the model, alternative approaches become necessary.
The volume of data produced from certain sources could also make unfeasible a centralized data collecting, in terms of money and computational efficiency, while introducing a single point of failure on the server.

1.2 Introducing Federated Learning

Federated learning (FL) is a new breed of AI that enables Machine Learning on **distributed** data across different systems, by **decentralizing** the training process on the device, where distributed data is located (Figure 1).
It is now possible to train excellent medical AI models by enabling different hospitals to work together. It is even possible solve financial fraud by training AI models on the data from different financial institutions. Moreover, building novel privacy-enhancing applications (such as secure messaging) that have better built-in AI than their non-privacy-enhancing alternatives is now feasible. Those are just a few of the examples.
The deployment of Federated Learning has revealed an increasing number of areas that can be deeply transformed due to accesses to vast amounts of previously inaccessible data.

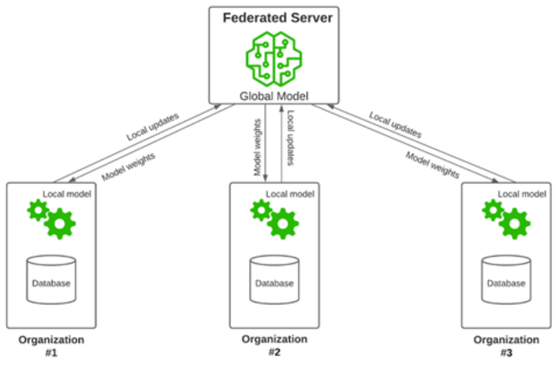

Figure 1: Generic Federated Learning architecture

1.2.1 Typical Federated Learning pipeline

As stated in [25] and in the official documentation of Flower Framework[1], at the high level an FL workflow can be resumed in:

1. Initializing the model parameters on the server and sending them to the edge devices (client nodes).
2. Training the model locally inside every decentralized device on its own set of data, typically not until full convergence but stopping at an early stage.
3. Sending the local parameter updates from the client nodes to the central server for aggregation, they can be just the gradients or the whole set of parameters.
4. Centrally aggregating the collected values, appropriately weighting them, utilizing techniques as Federated Averaging (*FedAvg*)
5. Repeat the process described from point 1 to point 4 until the model converges.

The update of the shared model can be expressed by the following expression:

$$M_{t+1} = M_t + \frac{1}{N}\sum_{k=1}^{N} u_t^k$$

where M_t denotes the shared model at the tth iteration, and u_t^k indicates the update from the kth client at iteration t.

[1]`https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html`

2 Threats and vulnerabilities in FL

As presented in [28], [19], and [21] an essential step to comprehending attacks with FL methodology is to acknowledge the threat model that assists us in detecting and describing possible security risks. A FL system offers a decentralized paradigm for processing ML tasks on an extensive scale while protecting user privacy and data. However, the creation of a federated system is a very challenging task in itself, as it is subject to several vulnerabilities that must be addressed. The system may be exposed to a variety of attacks targeting the federation of the system, network topologies, data partitioning, etc.
In this section, after a brief introduction, we will first assess the threat model of a FL system, and then explore the various sources of vulnerabilities.

2.1 The Setting of FL

As we briefly stated in section 1.2, a federated learning system consists of different devices performing ML algorithms cooperatively, under the assumption that private information of each participant is maintained local to it. The training phase is transferred to each client, performing training over the subset of local data it has: then, once training is complete, the client sends only parameter updates to the server, which only provides parameter aggregation to update the global model. This scheme aims to provide good training performance while granting protection for local client data.
In such a distributed environment, with many critical components, there are both communication overhead issues, and also hidden dangers of privacy leakage and tampering attacks.

2.2 Threat Model of Federated Learning

Let's start by identifying and enumerating potential threats and issues for a FL system. These threats will be further analyzed when we will talk about attacks to a FL system in section 3.

2.2.1 Security and Privacy Issues

One of the most critical factors when dealing with FL vulnerabilities and attacks is whether they pose a system security or a privacy risk. Depending on the application context, the repercussions of privacy attacks and data leakages may weigh much more than attacks to the system security or vice versa. For example, in the case of sensitive healthcare data (e.g. used for cancer detection models), privacy attacks may lead to critical patient data exposure, meanwhile in Intelligent vehicles, security attacks may lead to inaccurate commands, bringing both financial and personal damages.

2.2.2 Insider and Outsider Threats

Attacks to a FL system may be carried out by both:

- **Insiders**: insider attacks include those launched by the FL server, and by any client in the FL system.
- **Outsiders**: outsider attacks include those launched by eavesdroppers on the communication channel between the clients and the FL server, and by users of the final FL model when it is deployed as a service.

Typically, insider threats are considered more dangerous, as they highly enhance the capabilities of the adversaries:

- Attackers may compromise and get full control over one or more clients, and then be able in modifying valuable parameters, the local training procedure or the hyperparameters, or even the model's behaviour between each training round.
  In this case, the attacker may control the entire training process for a few participants, possibly compromising the integrity of the overall FL system.
- Another attack vector is the server itself: direct control of the server lead to attacks to the global trained model, potentially revealing more sensitive data.

Particularly, insider threats may lead to one of the following three general forms of attack:

- Single attack: a single, non-colluding malicious participant aims to cause the model to miss-classify a set of chosen inputs with high confidence;
- Byzantine attack: the byzantine malicious participants may behave completely arbitrarily, delivering arbitrary updates to the server that hinder the global learning model's efficiency and the convergence of FL models. Typically, byzantine participants tailor their outputs to have similar distribution as the correct model updates, making them difficult to detect;
- Sybil attack: because of its distributed design, federated learning is vulnerable to saturation attacks, or Sybil attacks, in which the adversaries can simulate multiple dummy participant accounts or select previously compromised participants to influence the global model's training, making the system susceptible to manipulation when servers lack detection capabilities.

On the other hand, outsider threats' main objectives are about sniffing data (e.g. Man-In-The-Middle attacks): exchanging model parameters may lead to confidentiality and integrity issues, and so securing and monitoring communication channels is mandatory.

2.2.3 Threats at Training and Inference Phase

During training phase, the main goals for attackers are to compromise the integrity of the training dataset (through data poisoning attacks), or rather to

compromise the integrity of the overall learning process (model poisoning attacks).
An attacker can also launch a range of inference attacks on an individual participant's update or on the aggregate of updates from all participants. At inference phase, the main objectives are not to tamper the overall FL model, but rather to cause it to produce wrong outputs or collect evidence about the model characteristics, since models at training phase will unintentionally memorize detailed information about private data.
The effectiveness is largely determined by the information that is available to the adversary about the model, and based on this we have different levels of threats.

- White-box attacks: the attacker has full access to the FL model.
- Black-box attacks: the attacker can only query the FL model.

FL requires extra efforts to defend against white-box attacks, since the model maintained by the server is made accessible to any malicious client during the previous training phase.

2.3 Vulnerabilities in Federated Learning

In the next paragraphs we will analyze the most critical vulnerabilities characterizing a FL system ([21] and [23]). Knowing and categorizing open and potential sources of vulnerabilities will help us to give a better definition of the requirements to grant security and privacy over the system and for the whole FL process life cycle.

2.3.1 Communication Protocols and Channels

A FL process implements iterative learning with randomly selected clients, requiring typically hundreds or thousands of rounds to reach convergence. This implies a significant amount of communication over the network between clients and the server, and therefore an insecure communication channel represents an open vulnerability.
A way to mitigate this is using Homomorphic Encryption (see section 4.4.2) which allows to protect clients' private data through model updates exchange between the clients and the server. Besides, it enables the server to perform computation on encrypted model updates without decrypting them.
Furthermore, potential communication bottlenecks may cause discarding of clients based on the connectivity status, which could lead to unwanted biases in the model and could weaken the model aggregation.
Moreover, a modern device can train on a batch of data in a few milliseconds, the generated model can be very large, even containing tens of millions of parameters (an example is a trained model of ResNet-50, which contains 23.5 million parameters, sized as 94MB [21]). Some devices have the capability of generating more updates than the network bandwidth could handle, moreover

in a typical FL environment, multiple devices that send periodic updates are connected to the same wireless network, leading to further congestion. These situations can result in accidental DoS, but there are techniques that mitigate this problem such as not sending the whole model as is but compressing it before updloading to the network, but then if the compression algorithm is lossy, the overall quality of the centralized model worsen.

2.3.2 Malicious Clients

While in traditional ML approaches clients simply provide training data, in a FL process clients represent a critical component of the overall architecture, since they observe intermediate snapshots of the global model and actively contribute to the updates at each round.
Malicious clients can either interfere with the training process, or rather exploit the global model parameters or training data they receive to craft attacks.

2.3.3 Gradient Leakage and Data Reconstruction

Although clients' data are not shared at training phase, it is shown that malicious clients are still able to reveal and reconstruct data and information about other clients' gradient updates, by saving snapshots of the consecutive global model parameters and computing the differences between them, to obtain aggregated updates from all participants [11]. Adversaries can take advantage of leaked gradients to infer valuable information about benign participants, and they can use those gradients to tailor their own updates such as they have similar distribution as the legitimate model updates, making them very difficult to be detected by defense tools.

2.3.4 Malicious Server and Aggregation Algorithm

Servers are responsible for sharing the initial model parameters, aggregating model updates, and communicating the global model to the selected clients, making it one of the most critical components of the FL system. A typical FL architecture presents several servers deployed in the cloud, which can be the target for DDoS attacks. Furthermore, a compromised server can observe all the gradient updates sent by clients at each training round and interfere with the aggregation process and the overall learning process. For these reasons, a server should be continuously monitored to detect any potential vulnerability.
This vulnerabilities can lead to various of the attacks that will be later described. These issues can range from inference actions (data extraction from clients, which are inference attacks) to attacks on the now exposed global model (model poisoning).
To be able to reconstruct private data of the clients, the malicious server can choose to only analyze the single updates from the clients (*passive attack*) or instead focus on isolating the individual models shared by the victims, aiming to more powerful (and *active*) attacks [20]. Another critical component to the

security of the FL system is the aggregation algorithm used by the server to compute the clients' updates and update the global model. For this reason, it represents the first line of defense against anomalies coming from malicious participants that are trying to damage the overall learning process.
Attacks like label-flipping (refer to 5.1) can produce devastating outcomes on the usability of the model if the aggregation algorithm is not robust enough or prevention measures are not taken.

2.3.5 Implementation and Deployment of the FL pipeline

Last but not least, the implementation process of the FL must be carefully considered: confusion and lack of understanding in what is considered to be sensitive user data and what is not can be one reason for the breach of security and privacy.

3 Taxonomy of Attacks in Federated Learning

3.1 Introduction

In this section, we are going to enumerate and model a taxonomy of attacks to privacy and security of FL systems, based on the threat model we developed in 2.2 and the vulnerabilities we analyzed in 2.3. Indeed, one of the most critical categories of attacks we considered for ML (and consequently, for FL systems) is adversarial attacks, where attackers aim to tamper the data and the overall model security. There are many different types of adversarial attacks, aiming at damaging:

- **Confidentiality (Privacy)**: attackers aim to retrieve confidential information about the FL model and about client private data.
- **Integrity**: attackers aim to manipulate the private data and the FL model accuracy.
- **Availability**: attackers aim to compromise the model's availability and the overall FL process through Denial of Service.

Let's now examine the most important categories of attacks to FL systems.

3.2 Attacks exploiting Data

In a general FL setting, a malicious attacker wants to exploit security vulnerabilities to gain control of either one or more clients or to the aggregator server, aiming to mislead the global learning model behavior. Typically, the largest exploitable threat surface in this setting is the cluster of clients participating in the training phase by contributing with data and computation. Based on the purpose of the attack, we can categorize security adversarial attacks into:

- **Poisoning Attacks**: these attacks aim to induce incorrect behaviour and model corruption either through manipulating clients' training data or the model updates sent by each client.
- **Backdoor Attacks**: these attacks aim to inject a malicious task into the existing model, to evade employed detection systems, while retaining the accuracy of the actual task.
- **Evasion Attacks**: these attacks aim to evade a deployed model by manipulating the data samples ("adversarial samples") fed into it.

3.2.1 Poisoning Attacks

FL clients have access to intermediate global model parameters, as well as the training set, and contribute to the overall training process. Poisoning attacks occur during the training phase, and can manipulate either the training set (data poisoning) or the training procedure of the local model, with the ultimate goal of

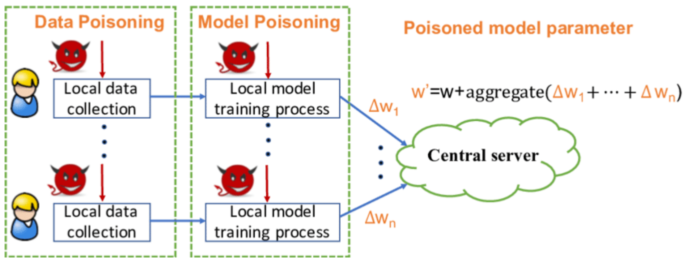

Figure 2: Data versus Model poisoning attacks in FL[2]

manipulating the global model itself (model poisoning). Either way, poisoning attacks try to modify the behavior of the target model in some undesirable way. Since model updates are taken from a large group of clients, consequently the probability for this kind of attacks to happen on one or more clients is very high, and so is its severity.
Let's now examine the two different types of poisoning attacks:

- **Data Poisoning**: in general, data poisoning is defined as the intentional generation of wrong and dirty samples to interfere maliciously with the behavior of the model. These kind of attacks can be further classified into dirty-label attacks and clean-label attacks: in the first case, an adversary can change the labels of any training data instance, while having access to the model and its parameters, meanwhile in the second case the adversary cannot alter ground-truth labels, but it can make targeted perturbations to features of the training data samples it wishes to misclassify to degrade the model performance. A classic example of a dirty-label poisoning attack is label flipping, first deployed over a classical ML model [2]: a malicious client can flip its training data labels belonging to a particular class to another different class, while keeping the data unchanged. This way, when it comes time to pass its local updates, the global model will incorrectly classify samples belonging to the flipped labels.

- **Model Poisoning**: in a model poisoning attack, malicious agents aim to alter the local model updates before sending them to the server, taking advantage of the clients' capability of influencing the federated global model. Unlike the traditional ML setting, a FL process is intrinsically vulnerable, without further defensive methods, to model poisoning, since the global model is federated for the entire training process, leading to numerous opportunities for the attackers to intercept the parameters sent to the clients. Furthermore, the effectiveness of model poisoning attacks inevitably increases when there is a large number of malicious clients. In general, a model poisoning attack is more effective than a data poisoning attack, as typically the malicious participant's updates are boosted and tailored to cause maximum damage to the global model's performance [7].

3.2.2 Backdoor Attacks

Backdoor attacks can be considered as a particular type of model poisoning: these attacks aim to inject and perform an hidden attacker-chosen task into the existing model, while retaining the accuracy of the actual main task to evade employed detection systems. A specific backdoor trigger is inserted at training time and this causes the model to misclassify when certain input patterns chosen by the attacker are fed into the model. A malicious participant may introduce a backdoor in the model by training it on benign training data as well as chosen backdoor data [17]. Evading anomaly detection is achieved into the training by using an objective function that both rewards the model for accuracy, and penalizes it for deviating from what the aggregator considers a "normal" result. Once poisoned, the backdoored model's weights are scaled up by a scaling factor, so to ensure that the attacker's contribution survives averaging and is transferred to the aggregated global model.

3.2.3 Evasion Attacks

Evasion attacks are a specific type of attack that aim to evade a deployed model by manipulating the testing data samples used after training phase [21], causing the ML model to misclassify observations (see figure 3). There are various types of misclassifications that can happen [16]: by reducing confidence score, by an attacker that makes the model classify an input to another (any) incorrect class (generic misclassification), by an attacker that produces a sample that make the model classify its inputs to a specific target class (target misclassification) and by an attacker that produces a sample that is itself classified as a target class (source/target misclassification). An attacker can have different knowledge of the system, three main scenarios can be identified: *white box*, in which the attacker has full knowledge of the classification model and training dataset, *black box*, in which the attacker has no knowledge on the target and has to try to find potential information leakages and *grey box*, which is an hybrid: the attacker knows something about the classification model but anything about the training dataset.

Evasion attacks can be very effective in FL environments, because the centralized model and the loss function are easily available for the attackers.
An example of a strategy that an attacker can pursuit is deriving the impact of each input feature on the final classification by recognizing trends in the data in which the main model is the most sensitive and crafting ad-hoc samples to classify certain samples belonging to a certain class to another.

[2]`https://www.researchgate.net/figure/Data-vs-model-poisoning-attacks-in-FL_fig2_347178320`

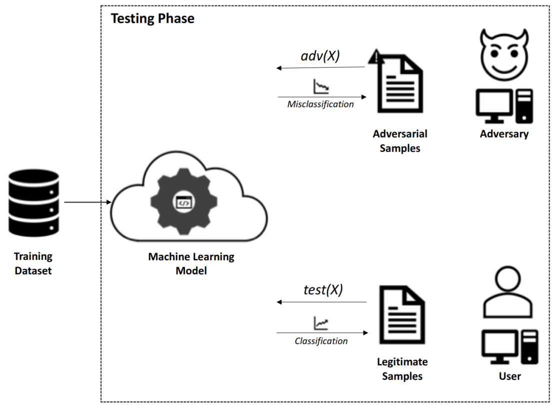

Figure 3: Model evasion attack diagram [16]

	Model Poisoning	**Backdoor Attacks**	**Evasion attacks**
Happens during	Training	Training	Testing
Goal	Damage the model to reduce performance	Trigger misclassifications under certain input patterns	Making the model misclassify specifically crafted inputs
Stealthiness	Low/medium, depends on how pervasive the model degradation gets	High, almost unnoticeable and dormant until activated	High, changes on the model are subtle and difficult to detect

Table 1: Comparison of attacks exploiting data

3.3 Attacks Focused on Federation

Since FL can occur over a set of diverse network topologies, settings and environments, creating a secure federated system is a challenging task due to the numerous attacks targeting the federation.

3.3.1 Inference Attacks

Inference attacks are a serious privacy issue, as they aim to infer training data details. For instance, as we already discussed, exchanging model updates during the learning process may result in gradient leakage, and could leak private information about features of clients' training data, such as class membership, class representations, and properties of other participants' data. Furthermore, it is also possible for an adversary to recover the original training data without expecting any prior knowledge about the training set: it has been shown that, by randomly initializing a sample and its label, and then feed them to the model and to compute the respective gradients, it is possible to recover original

training data by computing the distance between the original gradients and the dummy ones [15].
Among inference attacks, membership inference attacks aim to leak information by checking if a data sample belongs to the training set: in this case, the information on the training data set is inferred by training the predictive model to predict original training data.
A peculiar case of data leakage and reconstruction are represented by GAN-based inference attacks. A Generative Adversarial Network (GAN) is a network with two main components: a Generator, a network which increasingly learns to generate realistic training data, and a Discriminator, a classifier network which learns to distinguish the generator's fake data from the actual training data. The two networks are trained in competition with each other, the goal of the generator is to create data as realistic as possible misleading the discriminator, while the discriminator is trained to identify these fake data from the actual ones.

Let's briefly see how a GAN works:

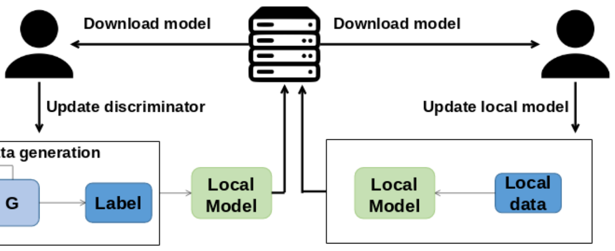

Figure 4: Overview of GAN-based poisoning attack in FL.[3]

- At training time, the Generator starts producing obvious fake data by taking random noise as its input (so to produce a wide variety of data), which is then transformed into a meaningful output. Then, the Discriminator uses the generated data and the real training data to compute a "Real/Fake" prediction, which is used to calculate the Generator's loss function (minimized if the prediction was wrong, increased otherwise).
- Likewise, at training phase, the Discriminator's loss function is computed by predicting an output for the Generator's fake data.

Attackers can train a GAN to generate samples without accessing the original data sample; consequently, many possible attacks can be involved with the use of a GAN. The fake samples can be used to compromise the global model by the update of a scaled version of these poisoned updates [14]. The GAN can be also used to perform an inference attack to reconstruct the private data samples of a client by model inversion.

[3]`https://www.researchgate.net/figure/Overview-of-GAN-based-poisoning-attack-in-FL_fig2_364164054`

3.3.2 Client unavailability

One of the FL assumptions is that participants don't drop the connection with the central server before sending updates, if many nodes decide deliberately to do so or get hampered there can be problems in the global model training like inefficiencies or unfairness.
One other main issue is that some clients may have important data that could yield to the formation of a more precise model, if those clients get down precious updates may be lost.

3.3.3 Free-riding attacks

Free-riding attacks are a peculiar type of attack that does not aim to degrade the performance of the system but instead is based on a correct FL workflow.
Free-riding clients are malicious clients that are interested in obtaining the correctly working model without actively participating in the iterative model update (either not sending updates or by sending artificially constructed weights with no actual value for the training). It is a passive attack, since malicious clients wait for the central model to be sent to them, the effectiveness of this attack is meaningful in the economic landscape, when data may be scarce and the model acquires monetary value.
It is also worth noting that the harmfulness on the global model degradation

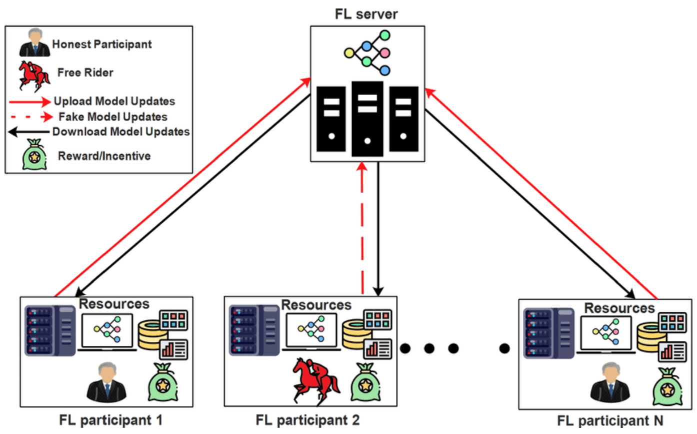

Figure 5: Free-riding attacks in the federated learning environment[4]

of these type of attack are inversely proportional to the dimension of the FL environment (refer to 3.3.2): when there are only a few nodes, the impact of a not cooperative one is relevant, but if there are many nodes this issue tends to fade. There can be two main reasons for free-riders to send useless updates: one is due to lack of collected data (or concerns about privacy) and the second is the willingness to save computing resources [10] and get benefits anyways. The methods for generating fake weights are various, they can be randomly

	Inference Attacks	**Client unavailability**	**Free-riding attacks**
Happens during	Training	Training	Training
Goal	Obtain information about the other clients and the original training set	Generate unfairness or lose precious model updates	Gain knowledge passively
Stealthiness	Medium, depending on how much the client's updates appear valid	Low/medium, very unreliable clients can be excluded from the training process with a proper defense	High, if fake updates are carefully crafted

Table 2: Comparison of attacks exploiting the Federation

extracted (and be typically easily detected as anomalies if some validation is applied by the central server on the received updates), they can be obtained without modifying the received ones or generated in more sophisticated ways. An example of more sophisticated weight generation is called delta weight generation [10] in which weights are obtained by the client subtracting two different version of the central model received at the end of previous updates (this type of generation is robust toward anomaly detectors because the submitted gradient is exactly the average weight update submitted by all clients in the previous round). Formalizing this, calling M_j the global model in round j and $G_{i,j}$ the gradient update sent by the client C_i in round j, then the update $G_{k,l}$ sent by the free-rider client C_k at round l is defined as

$$G_{k,l} = M_{l-1} - M_l = \eta \cdot \frac{1}{n} \sum_{c=1}^{n} G_{c,l-1}$$

with η being the learning rate of the client C_k. This strategy can be further improved by adding Gaussian noise to the delta-obtained weights, doing this guarantees that multiple free-riders clients will not have the same model updates at the end of each round, otherwise the probability of getting detected by centralized protection systems would be high and the impact of the attack would sharply decrease.

[4]`https://www.researchgate.net/figure/Free-riding-attacks-in-the-federated-learning-environment_fig7_356220574`

4 Defending Security and Privacy in Federated Learning

Defensive methods are deployed to protect the system against the known threat vectors and vulnerabilities, reducing the probability for attacks to exploit them. We can distinguish two types of defenses, namely proactive and reactive: a proactive defense implies the deployment of cost-effective methods in a production environment to figure out the threats and related risks prematurely, while a reactive defense by identifying the occurring attack and taking responsive measures, as part of the mitigation process. In this section, we will focus on an overview of the most frequently employed proactive defensive methods, giving a qualitative analysis of their effectiveness and limitations.

4.1 Anomaly detection and robust aggregation

Anomaly detection is a proactive defense technique which explicitly detects malicious clients' updates, preventing their impact on the global model. One approach to detect malicious updates is to measure the test error for each given update, and exclude those which do not contribute to (or, even worse, are aimed at tampering) the aggregated model. Many approaches have been successfully deployed using different methods, such as Krum, a majority-based algorithm combined with the minimization of the squared distance, used to expose any deviations for model updates and detect the presence of Byzantine clients [4]. Another method is given by AUROR [3], a statistical mechanism employing the K-Means algorithm to classify clients based on the indicative features (i.e. the updates) they uploaded: these features are used to cluster clients into benign and malicious groups, excluding the latter from the process.

4.1.1 Client reputation

A system based on assignment of a reputation score to each client can also be deployed: at aggregation time, each client is given a different weight based (also) on their reputation score, which is a measure of how much that particular client has been trustworthy during the previous N rounds of training, with N that can be adjusted to each task's necessities. Of course, a poor reputation score results in a low influence on that aggregation round, meanwhile a good score will make those client's updates more relevant.

4.2 Pruning and Fine-Pruning

Pruning is a general technique used to reduce a ML model's size, lowering the complexity and improve the overall accuracy. In FL, pruning is used as a proactive countermeasure to backdoor attacks [6]: it has been shown that, in a backdoored network, there are neurons which activate only in the presence of certain backdoored inputs, and remain dormant when clean inputs show up [8]; these neurons are used to recognize backdoors and consequently have the network misbehave. In this context, an effective pruning technique has been used to prune neurons based on an average activation measure. The same work highlights a category of pruning-aware attacks, where the malicious participant itself operates pruning of the network, followed by iterative small de-pruning steps to have the model learn the backdoor inputs; then, the pruned backdoored model is re-instated with the original pruned and dormant neurons (as the attacker cannot modify the model's hyperparameters): this way, the dormant neurons act as "decoy" to the defender's pruning. As a countermeasure, defenders of the FL system may employ pruning along fine-tuning ("fine-pruning"), by adding a small local retraining phase on a clean training dataset on the server, after the pruning step. This has shown very effective results in reducing the backdoor task accuracy.

4.3 Adversarial Training

Adversarial training (AT) is a general method whereby a ML model is trained by both clean training data and generated adversarial samples to gain robustness over adversarial attacks. AT is performed by iteratively solving a minimax optimization problem: first, the classification loss is maximized and the adversarial samples are generated, then the expected loss is minimized over the adversarial examples, and the model parameters are obtained. In FL, this is performed by a distributed solution of the problem, followed by parameter aggregation to obtain a robust global model: the crucial point is the aggregation algorithm, which must consider the non-i.i.d. of federated data, to have good accuracy performance and reduce the communication required for convergence. The work at [24] considers these issues and provides an efficient algorithm for adversarial training over non-i.i.d. data. Novel solutions employ GANs at each benign client nodes to generate fake images (having their visual features obfuscated), which will be mixed to the original training set and used to train the local model: this has proven to reduce the plausibility of images reconstructed by an attacker's GAN model while having minimal impact on the accuracy of the global model [27].

4.4 Privacy Protection Techniques

Based on those presented in [23] and [26], we give a taxonomy about the basic requirements regarding information privacy protection that must be guaranteed in FL systems, and traditional techniques used to satisfy them.

4.4.1 Secure Multi-Party Computation

Secure multi-party computation (SMC) is used to create methods for parties to jointly compute a function over their inputs, while keeping those inputs private. The key idea is to ensure that no party learns anything about the other parties' inputs apart from what can be inferred from the output. SMC enhances the privacy and security of this process by enabling secure gradient aggregation,

whereby model updates are securely aggregated without revealing individual updates, using secret sharing and distributed computation. There exists approaches [12] whereby secret sharing is employed by each pair of parties, i.e. each client and the server, to enable privacy-preserving FL using SMC. Several other works [9] show that integrating SMC with other techniques, such as client-level differential privacy and encryption can help in preventing data leakage of clients updates at the central server.
By SMC methodologies, data leakage is largely reduced by hiding intermediate computations, and robustness against inference attacks is achieved. The main disadvantage of this approach is the large communication overhead it introduces, growing as the number of parties increases, causing scalability issues.

4.4.2 Homomorphic Encryption

Homomorphic Encryption (HE) is an encryption technique which allows for algebraic operations to be performed directly over the cyphertext without prior decryption. The final results are therefore kept encrypted while applying transformations, and the decrypted result is the same as if those transformations were applied over the input plaintext (i.e. encryption and decryption functions are homomorphisms).
Depending on the specific encryption methods and applicable operations, we have Partially HE, where a single operation can be applied unboundedly, or Fully HE, where we have full capabilities over mathematical operations and on the number of times they can be applied. The main advantage of HE is that it does not require a decryption phase to apply operations and transformations on data, preventing attacks and enabling individual's privacy. In the context of FL, through HE we can enforce participants' privacy by having the centralized server performing algebraic operations directly on encrypted parameters without decryption. Proposals, such the one in [22], have implemented a multi-key HE, requiring participants collaborating to decrypt the results.
The main drawback in HE approaches is that the entire approach requires a high time cost as well as heavy computational cost during the encryption operation, which may also be a severe problem for low-powered edge devices participating in the FL process.

4.4.3 Differential Privacy

Differential Privacy (DP) is a privacy-preserving method used to enable individual user's privacy: a differentially private algorithm does not allow attackers to infer user's private information from its output[1]. In practice, DP techniques add random noise to the participants' uploaded sensitive parameters to protect each user's data, so that attackers cannot reverse the original training data used by clients.
The main advantage of DP is that the approach proposes privacy protection strictly independent of the background knowledge of the model we have. However, due to the random noise introduced into the parameters, these solutions

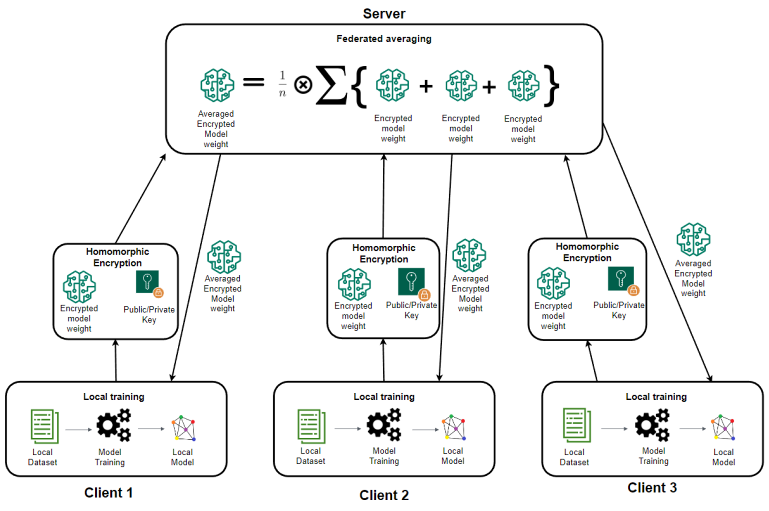

Figure 6: Homomorphic Encryption in a nutshell[5]

may bring uncertainty and loss of performance to the model, also making it harder for the server to evaluate the client's behaviour. DP has been combined with other methods, being SMC and HE, to secure FL models with a decreased noise factor [13]. DP approaches have been deployed as proactive countermeasures to inference and also evasion attacks, as addition of noise limits the ability of attackers in manipulating test samples.

4.4.4 Trusted Execution Environment

A Trusted Execution Environment (TEE) is generally defined as a secure high-level trusted ecosystem (composed of CPU and dedicated memory) in which code can be run securely. In a FL system, a TEE can be used for proactive privacy-preserving purposes in models, by allowing execution of gradient computation (on each client) and parameter aggregation (on the server) in an isolated area, while granting an authenticated communication channel between each participant and the server, along with authentication, confidentiality, and both local and remote attestation of the executed code, whereby a local or remote process can verify its trustworthiness [18].

[5]`https://www.mdpi.com/cryptography/cryptography-07-00048/article_deploy/html/images/cryptography-07-00048-g001.png`

5 Implementing Attacks to a FL system

In this section, we will show three different implementations for attacks previously examined in section 3. In particular, for the first two attacks, we will consider a FL system analyzing data coming from a dataset, consisting of acceleration samples collected from six sensors mounted on an industrial machine during manufacturing. The scenario presents a centralized server, which performs model initialization and parameter aggregation, and 6 clients, performing local training and sending intermediate updates to the server.
While the third attack will deal with a well-known dataset, the MNIST, to show the performance of a GAN-based inference attack on images. We will first explain the implemented attacks over the system, and then measure the accuracy performances of the system with and without an ongoing attack.

5.1 Label Flipping Attack

As we briefly saw in section 3.2.1, Label Flipping is a common example of a dirty-label data poisoning attack: adversarial participants start by deliberately changing the labels of their local training data to mislead the aggregated global model. The main objective for the malicious client is to send corrupted updates to the central server, which, if not mitigated, can degrade the performance of the global model, particularly on specific tasks if the attack was targeted to a specific subset of labels.
For our case scenario, we consider a set of malicious clients, and have them flip labels during the training phase, such that the overall global model gets compromised. In particular, we aim at performing a label flip between the one-hot encoded labels for samples representing two different materials manufactured by the IoT machines, representing aluminium and steel (respectively, label '1' and '2') tampering the malicious client's local model. Below, we have a procedure which performs label flipping during the client training phase:

Algorithm 1 Label Flipping

```
Input: $D^{train}$: training data, $y^{train}$: training labels, $N$: number of epochs
Output: $y^{flip}$: flipped one-hot encoded labels
Initialization: $y^{flip} \leftarrow y^{train}, \texttt{i} \leftarrow 1, \texttt{t} \leftarrow null$
1: procedure LabelFlipping
2:     for each label $y_j^{flip}$ do
3:         $\texttt{t} \leftarrow y_j^{flip}[1]$
4:         $y_j^{flip}[1] \leftarrow y_j^{flip}[2]$
5:         $y_j^{flip}[2] \leftarrow \texttt{t}$
6:         $y_j^{train} \leftarrow y_j^{flip}$
7:     end for
8: end procedure
```

5.1.1 Attack Performance

We have evaluated the aggregated accuracy performance for the model when it is under a label-flipping attack and when it is not. We considered different case scenarios, with different number of malicious participants to the FL system: we show the results considering one, three (50% of the total) and six (100% of the total) malicious participants. Below, we can see a plot summarizing the results we found:

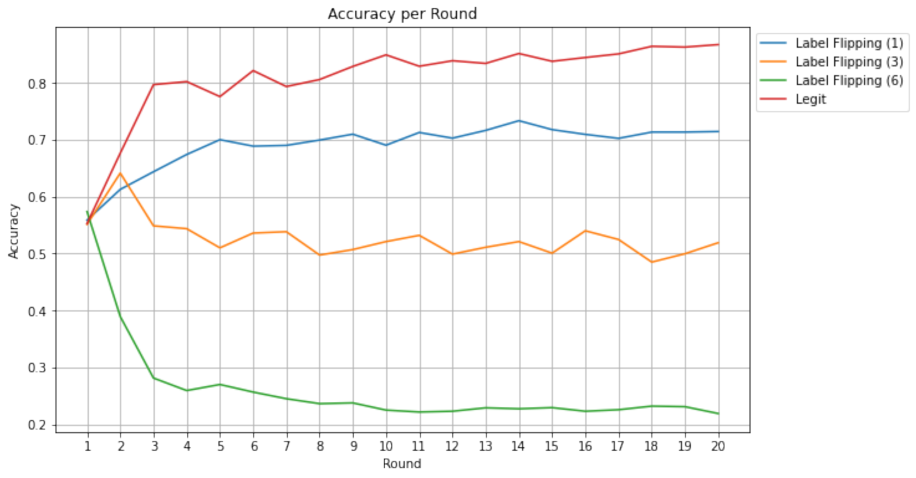

Figure 7: Label Flipping: Accuracy Measurements

As expected, the more adversarial clients we have, the more the aggregated accuracy decreases. Even with a single malicious participant, we can see that the accuracy is decreased approximately by 16.8%; the accuracy gets significantly worse when considering a system with 3 and 6 malicious clients, with an approximate decrease of 42.2% and 74.7% approximately.

5.1.2 Mitigation and Defense Solutions

Even if it presents large effectiveness in causing misclassification and vastly degrading the accuracy of the global model, the label flipping attack we presented remains susceptible to data certification and sanitization processes performed by clients at training time. An effective defense method employed to mitigate effects of label flipping is adversarial training: by training the model over generated adversarial samples, the model gains robustness against this category of poisoning attacks. Finally, a largely effective proactive countermeasure is to employ robust aggregation algorithms along with anomaly detection performed by the server, to effectively expose the deviations in the updates and exclude the malicious clients.

5.2 GAN-based Poisoning Attack

Next, we developed a poisoning attack by employing a Generative Adversarial Network (GAN): we recall that a GAN is composed by a Generator (G), employed to generate fake training inputs, and a Discriminator, used to distinguish and classify the input generated as fake/real. The loss function value for G is based on the output provided by D, while the loss value for D is determined by its capabilities of recognizing fake data. There are several works which detailed how GANs can be employed both for adversarial cases and defensive solutions. Particularly, following the idea behind the work at [29], we will shape our attack such that the trained generator is used to poison the local malicious model by employing a fake poisoned dataset, to undermine the global model performance.

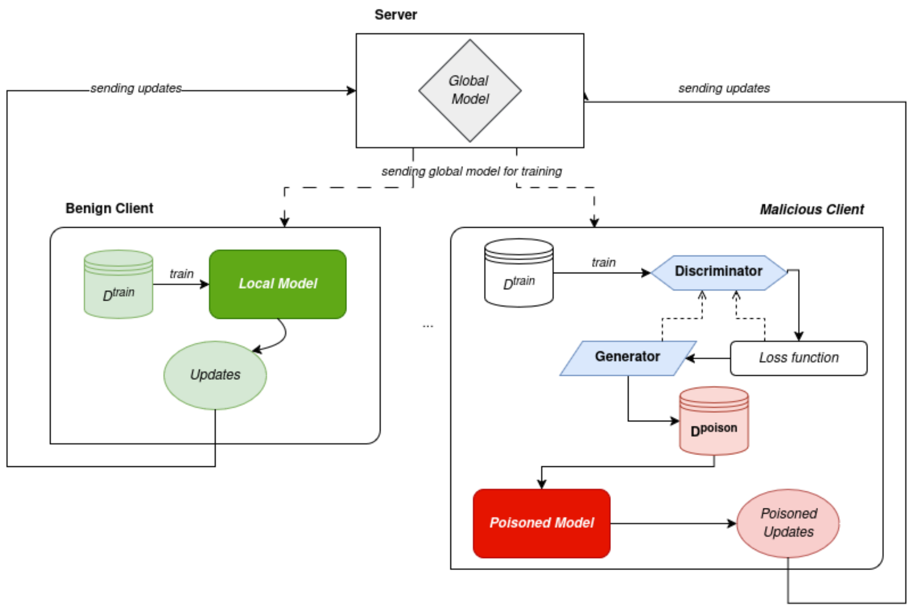

Figure 8: GAN-based Poisoning Attack

For our case scenario, we developed a generator and a discriminator model, both compliant to the IoT dataset we have on each participant, while keeping the CNN model already deployed over each client untouched. During the attack, the main goal for the adversarial G is to generate fake samples at each epoch, which will be given as input to D to compute the loss of G and increase its performance in generating finer fake samples. This basic algorithm provides the training phase for our GAN: the generator takes random noise as input and produces fake data D^{poison}, which is passed to the discriminator, which is also being trained on the real dataset, to make predictions and to compute its loss function and update its parameters. The generator's loss is then computed by using the discriminator's predictions for fake samples. After the GAN training phase, we start the local training of the model, received by the server, on the poisoned dataset D^{poison}, obtaining a malicious update to aggregate to the global model. Below, we show a summarized pseudo-code implementation of the attack:

5.2.1 Model Performance

Also for this attack, we have evaluated the aggregated accuracy performance for the model, both when it is under a poisoning attack and when it is not. We considered different numbers of malicious clients, training a GAN to poison the global model: we show the results considering one, three and six malicious participants. Below, we can see a plot summarizing the results we found: As

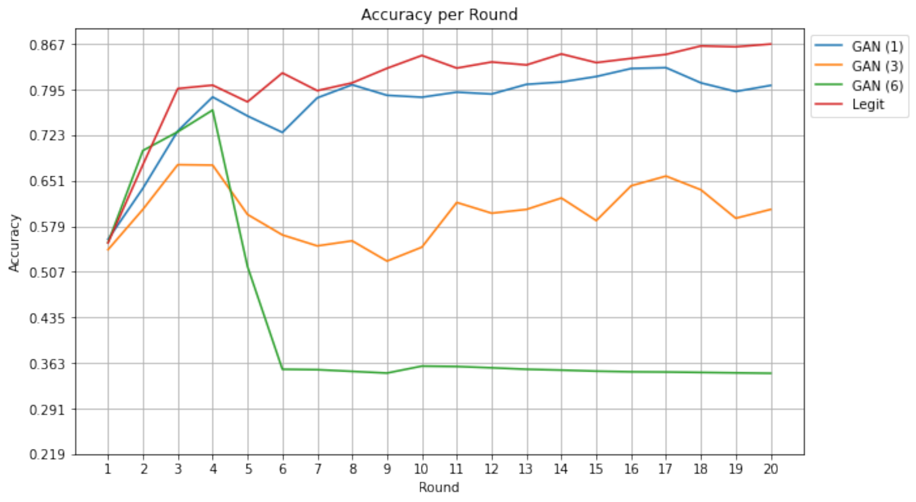

Figure 9: GAN-based Poisoning Attack: Accuracy Measurements

expected, the more adversarial clients we have, the more the aggregated accuracy decreases. A single malicious client training with a poisoned training set can decrease the accuracy approximately by 8.9% compared to a system with all benign participants; then, the accuracy measurements gets much lower when considering 3 and 6 malicious clients, with an approximate decrease of 31.9% and 60.1% approximately.

5.2.2 Mitigation and Defense Solutions

From the performance evaluations we measured, we can see that the GAN-based poisoning attack has a slightly lower impact on the accuracy of the aggregated global model, rather than the label flipping attack previously shown. However it is important to highlight that the GAN-based attack exploits generated data, typically presenting statistical characteristics similar to those of

Algorithm 2 GAN-based poisoning attack

Input: Client c_j's local training data: D_j^{train}, Number of training rounds for the GAN: N_{gan}, Number of training epochs N, learning rate η
Output: Poisoned training data: D_j^{poison}, local model parameters $\theta_{t,j}^{poison}$
Initialization: Deploy the GAN model at controlled client c_j

Phase 1 – Training the GAN

```
 1: procedure GanTraining
 2:     for each round $N$ do
 3:         $Out^{real} \leftarrow$ training $D$ on $D_j^{train}$ //$D$ predict for real dataset
 4:         $L_D^{real} \leftarrow loss(Out^{real}, l^{real})$
 5:         $D_j^{batch} \leftarrow G()$ // Generate fake data batch
 6:         $Out^{fake} \leftarrow D(D_j^{batch})$ // $D$ predict for fake generated data
 7:         $L_D^{fake} \leftarrow loss(Out^{fake}, l^{fake})$
 8:         $L_D \leftarrow L_D^{fake} + L_D^{real}$
 9:         $W_D \leftarrow W_D - \eta \cdot \nabla L_D$ // Update parameters for $D$
10:         $L_G \leftarrow loss(Out^{fake}, l^{real})$
11:         $W_G \leftarrow W_G - \eta \cdot \nabla L_G$ // Update parameters for $D$
12:         Include $D_j^{batch}$ into $D_j^{poison}$
13:     end for
14: end procedure
```

Phase 2 – Training the model

```
15: procedure TrainingPhase
16:     for each round $N$ do
17:         Receive global model $\tilde{\theta}_t$
18:         $Out^{train} \leftarrow$ training $\tilde{\theta}_t$ on $D_j^{poison}$
19:         $L \leftarrow loss(Out^{train}, l^{train})$
20:         $W \leftarrow W - \eta \cdot \nabla L$
21:         Obtain $\theta_{t+1,j}^{poison}$ and upload it to the server
22:     end for
23: end procedure
```

the original training set, resulting in a more effective and stealthier attack than label-flipping: it is however important to carefully evaluate the *effectiveness-stealthiness* trade-off to obtain a well-executed attack [29]. Other mitigation methods such as adversarial training remain effective to gain robustness from poisoned datasets.

5.3 GAN-based Inference Attack

The capacity of GAN to generate samples that appear to come from the training set without having access to the real dataset, makes it a promising technique for inference attacks. In this section, we present a GAN-based inference attack that aims to reconstruct private data samples of a client by model inversion, based on [5]. In particular, this attack shows how federated learning cannot protect the privacy of training sets of honest participants, exposing them to malicious actors.

As exposed in [5] this attack can be considered as a **privacy violation** is several cases:

- A victim device contains medical records of patients with cancer. The GAN is capable of generate data with the same distribution of those in the training set.
- The victim's device contains private images. The GAN will generate similar scenes resulting in an information leaked that reveals sensible informative about that person.
- The victim's device contains speech recordings. The GAN will generate a lot of word-like sounds, from which it may be possible to infer language used and whether the speaker is a male of a female.

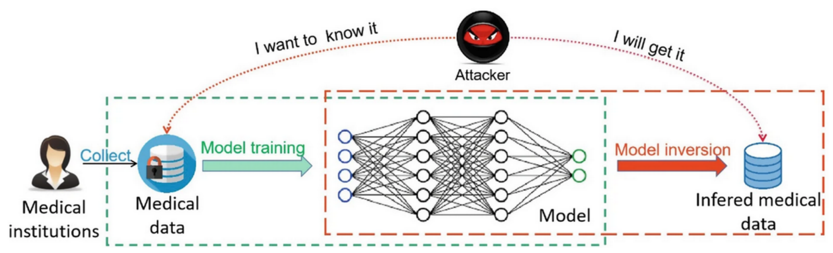

Figure 10: Model inversion attack in Federated Learning[6]

5.3.1 Attack description

To perform the attack, we considered a malicious actor that pretends to be an honest participant in the federated learning process, and another participant

[6]`https://link.springer.com/chapter/10.1007/978-981-19-8692-5_2/figures/2`

that is the victim. The objective of the attacker is to extract information about a specific class of data he does not own by manipulating the federated model and building a GAN locally. To enhance the effectiveness of the attack, the malicious actor will also actively influence the learning process to prompt the victim into releasing further details about the targeted class.
The attacker's behavior involves the use of a **Generator network** to create fake samples, aiming to make them closely resemble the private data of the victim's targeted class. To understand whether these generated samples mimic the target class, the attacker utilizes the federated model, which has been trained using the victim's private data, and use the output to compute the loss function for the generator. This loss measures how closely the model's predictions for the generated samples align with the characteristics of the target class. Then, the attacker uses these synthetic samples to further train the federated model, intentionally mislabeling them with incorrect class labels.
The steps performed by the attacker can be better shown with the following pseudo-code:

Algorithm 3 GAN-based inference attack

```
1: The federated learning system has at least 2 participants: an attacker (A)
   and a victim (V)
2: V has a training set with samples of classes $\{c_0, \dots, c_n\}$
3: A has a training set with samples of classes $\{c_1, \dots, c_n\}$
4: A wants to create fake instances of class $c_0$
   Phase 1 – Fake samples generation
5: for each round do
6:     download the parameters of the global model from the server and update
   the local model.
7:     generate fake samples with G
8:     use the local model to compute the probability for each fake sample to
   be classified of class $c_0$
9:     use the predicted probabilities to compute the loss function for G
   Phase 2 – Federated model training
10:    label all the samples generated by G with classes $\{c_1, \dots, c_n\}$
11:    train the local model with these samples
12:    upload the parameters of the local model to the server
13: end for
```

5.3.2 Attack Performance

We conducted our tests using the MNIST dataset, which consists of 32x32 grayscale images of handwritten digits from 0 to 9. The dataset is composed by 60,000 training data records and 10,000 test data records.
The victim's training dataset includes samples from all classes (digits 0 to 9),

whereas the attacker's dataset contains samples from all classes except one, which remains private to the victim. Figure [11] show two different tests: one where the client's private class is digit 3, and another where it is digit 9. The top rows show the training samples private to the victim, while the bottom rows show the evolution of the fake samples generated by the attacker's GAN over the different rounds.

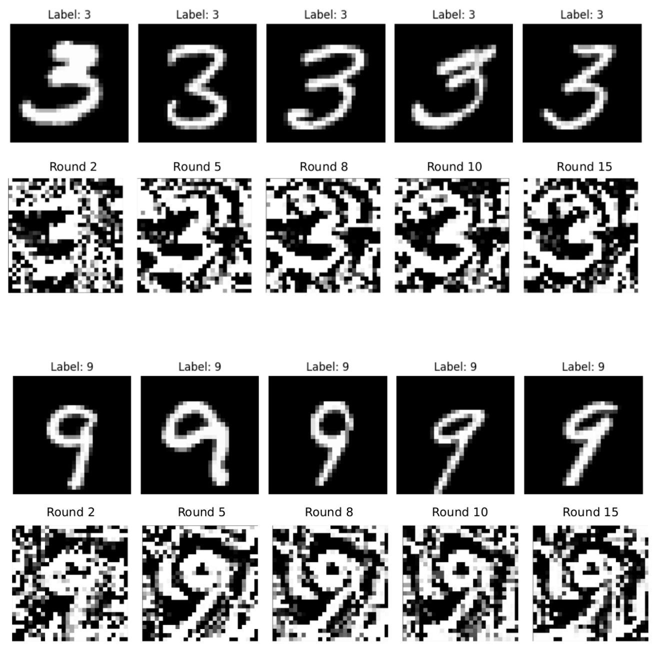

Figure 11: Generated samples with the GAN-based inference attack

What we show with this attack is that the GAN can be used to create instances of a class that is supposed to be private. A malicious participant can influence other honest participants and force them into releasing relevant information about their private datasets based on how the model evolves.
It is also important to highlight the **limitations** of this attack. The primary drawback is its reduced effectiveness in large-scale federated learning scenarios involving several clients. This is because the GAN-based attack can only infer class-wised representatives, which are generic samples characterizing class properties rather than the exact samples. As a result, the attack cannot provide any specific information about an individual target client. This limitation makes it unsuitable for situations where the goal is to target a specific client rather than an entire class of data.
Moreover, in some cases there is the risk that the attacker might think he has recovered sensitive information when he is just getting meaningless information. However, extensive research in the ML community has shown that GAN generated samples are quite similar to the training data, thus the results coming

from our attack reveal accurate information.

5.3.3 Mitigation and Defense Solutions

Defense mechanisms, such as obfuscating uploaded parameters via differential privacy, have a limited effectiveness against GAN-based inference attack.
A valid countermeasure against these attacks is using **adversarial training**, introduced in section [4.3], with a mechanism called Anti-GAN proposed in [27]. The idea of this mechanism is that the victim generates fake samples X' by training a GAN model using the private dataset X. Then, the generated X' is mixed with X to form the training dataset $\hat{X}$, which is used to train the federated model. To mitigate the data leakage risks, but preserving the federated model accuracy, the visual characteristics of the generated images (e.g., edges and corners) are obfuscated through window-wise transformations, while their classification features (e.g., image patterns) remain similar to the real dataset.

6 Conclusions

In this work, we examined the concept of Federated Learning (FL), highlighting its advantages and the significant gaps it has when compared to traditional, centralized machine learning models. We found that Federated Learning offers notable privacy benefits, as it allows end users to keep their data on their local devices rather than transmitting it to a central server, following a heavily decentralized approach.
However, this same decentralization also introduces new vulnerabilities, which expose the infrastructure to a range of novel threats.
A key challenge identified is the potential for malicious clients to exploit various vulnerabilities in the system. These vulnerabilities can lead to data leakage, model poisoning, or even a general degradation of the global model's performance. The communication protocol and the channels used for data transfer are critical points of weakness, and we explored how these factors contribute to the overall security risks in FL systems.
We then tried to assess this new knowledge in a practical scenario, evaluating how different attacks lead to which outcome. The first attack analyzed was Label Flipping, a type of attack that significantly reduces the accuracy of the global model by altering the labels of the training data. While this attack can cause substantial impairment in model performance, it is relatively easy to detect and counter with basic defensive measures on the server side.
Building on this, after discovering how Generative Adversarial Network (GAN) are promising in these applications, we employ them in a GAN-based Inference attack and in a GAN-based Model Poisoning attack: the results were consistent with our expectations (a general model degradation in the Model Poisoning and a particularly good label inference of certain classes within the MNIST dataset).
In conclusion, Federated Learning emerges as a highly promising technology that can provide a substantial increase in privacy for edge devices, as it allows sensitive data to remain localized rather than being sent to a central server. While this approach offers clear advantages in terms of privacy, it also introduces new security challenges that must be addressed. When appropriate defensive measures are implemented at the server level, such as mechanisms to detect and mitigate attacks, Federated Learning can continue to provide a high level of privacy without significantly compromising model performance.
It is clear that this technology holds considerable potential and will likely see increased deployment and refinement in the years to come, as its advantages in privacy and decentralization become more critical in a world increasingly focused on data security.

References

[1] Cynthia Dwork. “Differential Privacy”. In: *Automata, Languages and Programming*. Ed. by Michele Bugliesi et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12. ISBN: 978-3-540-35908-1.

[2] Battista Biggio, Blaine Nelson, and Pavel Laskov. *Poisoning attacks against support vector machines*. Mar. 2013. URL: `https://arxiv.org/abs/1206.6389`.

[3] Shiqi Shen, Shruti Tople, and Prateek Saxena. “Auror: defending against poisoning attacks in collaborative deep learning systems”. In: *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACSAC ’16. Los Angeles, California, USA: Association for Computing Machinery, 2016, pp. 508–519. ISBN: 9781450347716. DOI: `10.1145/2991079.2991125`. URL: `https://doi.org/10.1145/2991079.2991125`.

[4] Peva Blanchard et al. “Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: `https://proceedings.neurips.cc/paper_files/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf`.

[5] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. *Deep models under the Gan: Information leakage from Collaborative Deep Learning*. Sept. 2017. URL: `https://arxiv.org/abs/1702.07464`.

[6] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. “Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks”. In: *Research in Attacks, Intrusions, and Defenses*. Ed. by Michael Bailey et al. Cham: Springer International Publishing, 2018, pp. 273–294. ISBN: 978-3-030-00470-5.

[7] Arjun Nitin Bhagoji et al. “Analyzing Federated Learning through an Adversarial Lens”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 634–643. URL: `https://proceedings.mlr.press/v97/bhagoji19a.html`.

[8] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. *BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain*. 2019. arXiv: `1708.06733 [cs.CR]`. URL: `https://arxiv.org/abs/1708.06733`.

[9] Meng Hao et al. “Towards efficient and privacy-preserving federated deep learning”. In: *ICC 2019-2019 IEEE international conference on communications (ICC)*. IEEE. 2019, pp. 1–6.

[10] Jierui Lin, Min Du, and Jian Liu. “Free-riders in Federated Learning: Attacks and Defenses”. In: *CoRR* abs/1911.12560 (2019). arXiv: `1911.12560`. URL: `http://arxiv.org/abs/1911.12560`.

[11] Luca Melis et al. "Exploiting Unintended Feature Leakage in Collaborative Learning". In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 691–706. DOI: 10.1109/SP.2019.00029.

[12] Vaikkunth Mugunthan et al. "SMPAI: Secure Multi-Party Computation for Federated Learning". In: 2019. URL: https://api.semanticscholar.org/CorpusID:220598116.

[13] Stacey Truex et al. "A Hybrid Approach to Privacy-Preserving Federated Learning". In: *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. AISec'19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 1–11. ISBN: 9781450368339. DOI: 10.1145/3338501.3357370. URL: https://doi.org/10.1145/3338501.3357370.

[14] Jiale Zhang et al. "Poisoning Attack in Federated Learning using Generative Adversarial Nets". In: *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. 2019, pp. 374–380. DOI: 10.1109/TrustCom/BigDataSE.2019.00057.

[15] Ligeng Zhu, Zhijian Liu, and Song Han. "Deep Leakage from Gradients". In: *Advances in Neural Information Processing Systems*. 2019.

[16] Md. Ahsan Ayub et al. "Model Evasion Attack on Intrusion Detection Systems using Adversarial Machine Learning". In: (2020), pp. 1–6. DOI: 10.1109/CISS48834.2020.1570617116.

[17] Eugene Bagdasaryan et al. "How To Backdoor Federated Learning". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, Aug. 2020, pp. 2938–2948. URL: https://proceedings.mlr.press/v108/bagdasaryan20a.html.

[18] Yu Chen et al. "A training-integrity privacy-preserving federated learning scheme with trusted execution environment". In: *Information Sciences* 522 (2020), pp. 69–79. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2020.02.037. URL: https://www.sciencedirect.com/science/article/pii/S0020025520301201.

[19] Lingjuan Lyu, Han Yu, and Qiang Yang. "Threats to federated learning: A survey". In: *arXiv.org* (Mar. 2020). URL: https://arxiv.org/abs/2003.02133.

[20] Mengkai Song et al. "Analyzing User-Level Privacy Attack Against Federated Learning". In: *IEEE Journal on Selected Areas in Communications* 38.10 (2020), pp. 2430–2444. DOI: 10.1109/JSAC.2020.3000372.

[21] Nader Bouacida and Prasant Mohapatra. “Vulnerabilities in Federated Learning”. In: *IEEE Access* 9 (2021), pp. 63229–63249. DOI: `https://doi.org/10.1109/ACCESS.2021.3075203`. URL: `https://ieeexplore.ieee.org/document/9411833`.

[22] Jing Ma et al. “Privacy-preserving federated learning based on multi-key homomorphic encryption”. In: *arXiv.org* (Apr. 2021). URL: `https://arxiv.org/abs/2104.06824`.

[23] Viraaji Mothukuri et al. “A survey on security and privacy of federated learning”. In: *Future Generation Computer Systems* 115 (2021), pp. 619–640. ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2020.10.007`. URL: `https://www.sciencedirect.com/science/article/pii/S0167739X20329848`.

[24] Devansh Shah et al. *Adversarial training in communication constrained federated learning*. 2021. arXiv: `2103.01319 [cs.LG]`.

[25] Daniel J. Beutel et al. *Flower: A friendly federated learning research framework*. Mar. 2022. URL: `https://arxiv.org/abs/2007.14390`.

[26] Jie Wen et al. “A survey on Federated Learning: Challenges and Applications - International Journal of Machine Learning and Cybernetics”. In: *SpringerLink* (Nov. 2022). URL: `https://link.springer.com/article/10.1007/s13042-022-01647-y`.

[27] Xinjian Luo and Xianglong Zhang. *Exploiting Defenses against GAN-Based Feature Inference Attacks in Federated Learning*. 2024. arXiv: `2004.12571 [cs.CR]`. URL: `https://arxiv.org/abs/2004.12571`.

[28] Ghazaleh Shirvani, Saeid Ghasemshirazi, and Behzad Beigzadeh. “Federated learning: Attacks, defenses, opportunities, and challenges”. In: *arXiv.org* (Mar. 2024). URL: `https://arxiv.org/abs/2403.06067`.

[29] Wei Sun et al. *A GAN-Based Data Poisoning Attack Against Federated Learning Systems and Its Countermeasure*. 2024. arXiv: `2405.11440 [cs.CR]`. URL: `https://arxiv.org/abs/2405.11440`.