Federated Learning under Adversaries: Taxonomy of Attacks and State-of-the-art Defenses for MLSecOps

Ayoub Bendraou Aya Fsahi Nazih Ouchta Salim Ghoudane

November 2025

Abstract

Federated Learning (FL) has emerged as a paradigm that enables collaborative model training across distributed clients without centralizing sensitive data, offering significant privacy and compliance advantages over conventional machine learning. However, its decentralized nature introduces new attack surfaces and complex adversarial behaviors across data, model, communication, and aggregation layers. This state-of-the-art (SOTA) report situates FL within the broader context of Machine Learning Security Operations (MLSecOps) by mapping the evolution of attack taxonomies and defense mechanisms. It presents an integrated overview of integrity threats such as poisoning and backdoor attacks, privacy risks including gradient inversion and membership inference, communication-layer exploits like Sybil and man-in-the-middle attacks, and aggregator-level vulnerabilities such as model replacement and free-riding. Building on these foundations, the work surveys SOTA defenses—ranging from Byzantine-robust aggregation rules (Krum, Bulyan, FLTrust) and adaptive anomaly detection to differential privacy, secure aggregation, and federated adversarial training. The analysis emphasizes the trade-offs between privacy, robustness, and scalability, highlighting emerging trends in layered, privacy-preserving defense frameworks and MLSecOps pipelines capable of anticipating and autonomously mitigating threats throughout the FL lifecycle.

Contents

1 What is Federated Learning?

Federated Learning (FL) is formally defined and described across the literature as:

- A machine learning setting where **many clients** (e.g., mobile devices or whole organizations) **collaboratively train a model** under the orchestration of a central server (e.g., service provider), while **keeping the training data decentralized** [1].
- A learning paradigm introduced in 2016 by McMahan et al. where the learning task is solved by a loose federation of participating devices (clients) coordinated by a central server [1].
- A mechanism that embodies the principles of **focused collection and data minimization** [1]. Client raw data is stored locally and is explicitly not exchanged or transferred [1]. Instead, **focused updates intended for immediate aggregation** are used to achieve the learning objective [1].
- A model that mitigates many of the **systemic privacy risks and costs** resulting from traditional, centralized Machine Learning (ML) and data science approaches [1].
- A **privacy-preserving decentralized approach**, which keeps raw data on devices and involves local ML training while eliminating data communication overhead [2]. A federation of the learned and shared models is then performed on a central server to aggregate and share the built knowledge among participants [2].

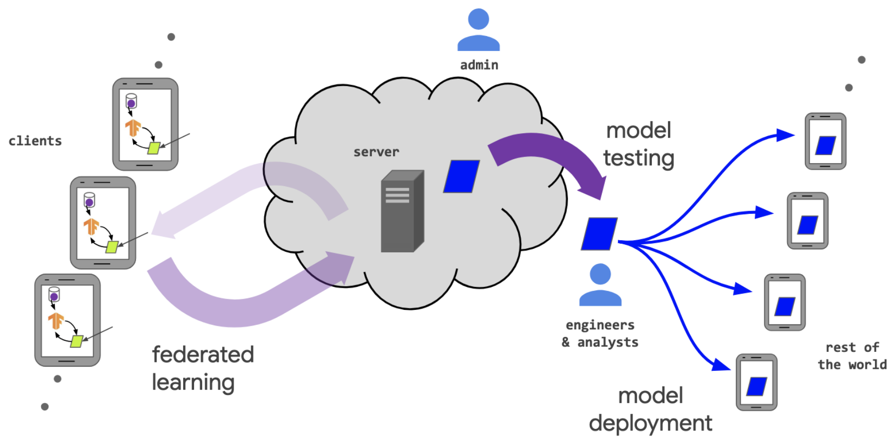

Figure 1: the various actors in a federated learning system [1]

1.1 Motives and Need for Federated Learning

The necessity for FL arose from the convergence of several technology trends and inherent failures within previous machine learning (ML) architectures, particularly regarding data privacy and scalability.

1.1.1 The Data Revolution and AI's Needs

The first motive is the **unprecedented availability of data** generated by connected devices, such as smartphones, Internet-of-Things (IoT) devices, and wearable medical devices . These modern mobile devices possess powerful sensors (cameras, microphones, GPS) and are frequently carried, giving them access to a **wealth of data** . Since ML models are **data hunger** and Deep Learning (DL) applications have experienced vigorous growth, models trained on this rich, real-world data hold the promise of **greatly improving usability** by powering more intelligent applications, such as improving speech recognition and text entry .

1.1.2 Centralized Learning Failures and Privacy Mandates

The conventional ML architecture relies on a **cloud-centric architecture** where data is continuously streamed, centrally stored, and processed . This centralized model faced severe limitations:

- **High Risk of Disclosure:** The **sensitive nature of the data** means there are risks and responsibilities to storing it centrally . Without serious privacy consideration, sensitive data is **highly exposed to disclosure, attacks, and cyber risks**. Breaches like Equifax, Marriott, and eBay demonstrated these risks in recent memory.

- **Regulatory Compliance Challenges:** Stringent regulations like the European Union's **GDPR** and the **HIPAA** guidelines strictly limit data sharing and storage. These acts challenge centralized models, as patient personal indicators cannot be shared without consent, meaning centralized AI-driven health analytics models are challenged with the quality of shared data (often anonymized, yielding generic interpretability).
- **Cost and Latency:** Centralized practices engage **unacceptable latency and high cost** .

1.3. On-Device Learning Isolation

While running ML models locally on devices (on-site ML) solves the data transfer problem, it creates a new drawback: the local models are limited to **each user's experience** and do not **benefit from peers' data** .

FL as the Solution: FL was developed to enable users to **collectively reap the benefits of shared models** trained from rich data, without the need to centrally store it . This approach allows the decoupling of model training from the need for direct access to the raw training data .

1.2 Mechanism: How does Federated Learning work?

The standard FL workflow is orchestrated by a central server and is often based on the **Federated Averaging (FedAvg) algorithm** [1, 2].

Architecture and Workflow

The FL life cycle consists of several continuous communication rounds until the global model reaches the desired accuracy [2].

1. **Client Selection/Initiation:** The server generates or initializes a generic model [2]. In each round, a subset of clients is sampled [1]. Typical eligibility requirements include the device being **in charge, idle, and on an unmetered connection** (i.e., Wi-Fi) [2]. Advanced protocols like **FedCS** may require clients to report information about their resources (upload and update time) to a Mobile-Edge Computing (MEC) server, which then determines the subset able to complete the FL steps within a deadline [2].
2. **Broadcast and Download:** Selected clients download the current model parameters/weights and a training program (e.g., a TensorFlow graph) from the server [1, 2]. The global updates might propagate to local nodes at different time instants depending on communication channel specifics [3].
3. **Local Training:** Each device locally trains and optimizes the global model using its local data [2]. This involves running SGD [2].
4. **Upload:** The client sends its locally computed parameters/updates to the central server [2].
5. **Aggregation:** The server collects and aggregates the updates. This process is often a **weighted average** based on client dataset size [2, 3].
6. **Model Update:** The server uses the aggregated gradient to update the global model. This new shared model is distributed back to the clients, and the process repeats until convergence [2, 3].

In this star network communication topology, a subset of selected devices performs local training on their **non-identically-distributed user data** and sends these local updates to the server [4].

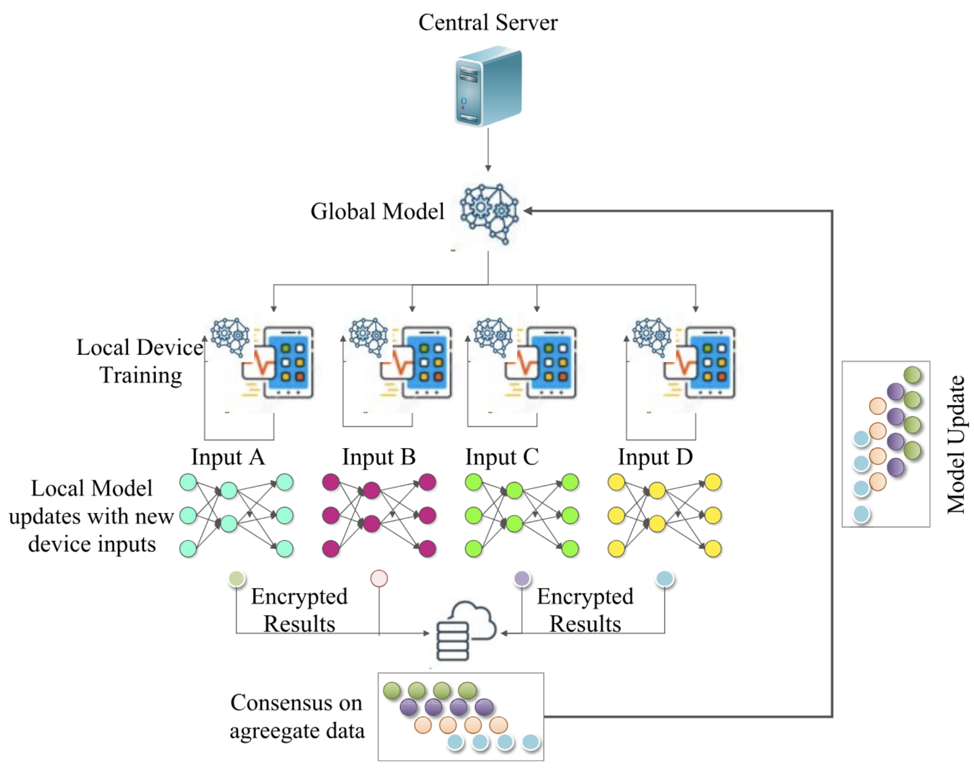

Figure 2: FL training process.

1.2.1 Alternative Learning Paradigms and Techniques

- **Fully Decentralized / Peer-to-Peer Distributed Learning:** This approach removes the need for a central server coordinating the overall computation, with devices communicating directly with neighbors [1].
- **FL Algorithms for Aggregation:**
  - **FedPer (Federated learning with personalization layer):** Clients send base layers (for representational learning) to the server but retain higher layers (decision specifics) locally [3].
  - **FedMA (Federated learning with matched averaging):** Constructs a global model by averaging values of hidden layers (e.g., convolutional layers) [3].
  - **FedDist (Federated Distance):** Computes the distance between neurons with similar features, suitable for sparse data [3].
  - **One-shot FL:** The massive number of communication rounds is substituted with only **one round** where each device trains its model until completion, and models are aggregated using ensemble learning techniques [2].
- **Data Partitioning:**
  - **Horizontal FL (HFL):** Local clients have the **same feature sets** but in **different data spaces** [3].
  - **Vertical FL (VFL):** Local clients have the **same sample space** but **different feature sets** [3].
  - **Transfer Learning-based FL (TL-FL):** Extends VFL to include more local clients with different feature sets but the same sample space, extracting common features to a shared space, suitable for EHRs collected from heterogeneous sources [3].

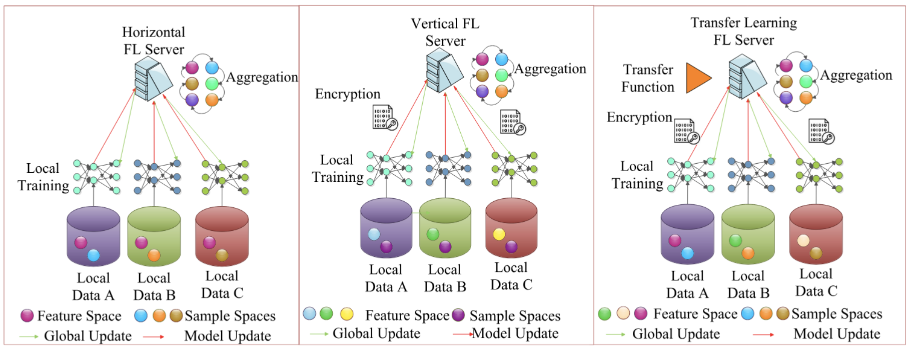

Figure 3: FL-HI aggregation conceptual overview: The left figure depicts the horizontal FL, centre figure depicts the vertical FL type, and the right figure indicates the transfer learning scheme in FL [3]

2 Technical overview

2.1 Federated Optimization Problem Formulation

FL focuses on supervised machine learning (ML), where the goal is to find model parameters $\mathbf{w}$ (a vector) that minimize a global loss function $F(\mathbf{w})$ over a massive, partitioned dataset.

The Global Objective Function (Non-IID Setting) The overall optimization goal is to minimize the finite-sum objective function $F(\mathbf{w})$, which is a weighted sum of the local loss functions $F_k(\mathbf{w})$ across K clients:

$$\min_{\mathbf{w}} F(\mathbf{w}), \quad where \quad F(\mathbf{w}) \overset{def}{=} \sum_{k=1}^{K} \frac{n_k}{n} F_k(\mathbf{w})$$

Where:

- K is the total number of clients participating in the learning rounds.
- n_k is the number of data samples on client k, where $n_k = |P_k|$ (P_k is the partition of data assigned to client k).
- n is the total number of samples across all clients, where $n = \sum_{k=1}^{K} n_k$.
- $F_k(\mathbf{w})$ is the **local objective function** (empirical risk) for the kth client, typically defined as $F_k(\mathbf{w}) = \frac{1}{n_k} \sum_{i \in P_k} f_i(\mathbf{w})$, where $f_i(\mathbf{w})$ is the loss function for the ith data sample (x_i, y_i).

2.2 The Federated Averaging (FedAvg) Algorithm

FedAvg is the chosen practical algorithm for FL implementation, combining local Stochastic Gradient Descent (SGD) on each client with a server that performs iterative model averaging. The algorithm is designed to use additional local computation to **reduce the number of communication rounds** by 10–100× compared to synchronized SGD.

The algorithm uses three key parameters to control computation:

- C: The fraction of clients selected to perform computation in each round.
- E: The number of local training passes (epochs) each selected client makes over its local dataset in one communication round.
- B: The local minibatch size used for client updates. ($B = \infty$ means the full local dataset is used as one minibatch).

The number of local updates per round for client k is $u_k = E\frac{n_k}{B}$.

FedAvg Pseudocode (Adapted from Algorithm 1) Server Executes:

```
1. Initialize global model weights $\mathbf{w}_0$.
2. For each round $t = 1, 2, \ldots, T$ do:
    • $m \leftarrow \max(C \cdot K, 1)$ (Number of clients selected).
    • $S_t \leftarrow$ (random set of $m$ clients).
    • For each client $k \in S_t$ in parallel do: $\mathbf{w}_k^{t+1} \leftarrow ClientUpdate(k, \mathbf{w}_t)$
    • Aggregate (Weighted Average): The new global model $\mathbf{w}_{t+1}$ is calculated as the weighted sum of the
      returned local models, where $\sum_{k=1}^{K} \frac{n_k}{n} = 1$.
          $\mathbf{w}_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{n} \mathbf{w}_k^{t+1}$
```

ClientUpdate(k, w): (**Run on client** k)

```
1. $\mathcal{B} \leftarrow$ (split local dataset $P_k$ into batches of size $B$).
2. For each local epoch $i$ from 1 to $E$ do:
    • For batch $b \in \mathcal{B}$ do: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \ell(\mathbf{w}; b)$
3. return $\mathbf{w}$ to server.
```

FedSGD as a Baseline

FedAvg generalizes **Federated SGD (FedSGD)**, which is the simplest synchronous update scheme. FedSGD corresponds exactly to FedAvg when $B = \infty$ (full local batch) and $E = 1$ (one local epoch).

- In FedSGD, each client k computes $\mathbf{g}_k = \nabla F_k(\mathbf{w}_t)$, the average gradient on its local data.
- The server aggregates the gradients and applies the update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{g}_k$.
- This is mathematically equivalent to the model averaging: $\forall k, \mathbf{w}_k^{t+1} \leftarrow \mathbf{w}_t - \eta \mathbf{g}_k$ and then $\mathbf{w}_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{w}_k^{t+1}$.

3 Applications

FL is deployed across massive consumer products (Cross-Device) and organizations with high confidentiality needs (Cross-Silo).

Consumer, Mobile, and Edge Devices (Cross-Device FL)

- **Mobile Keyboards:** FL was first tested on Google's virtual keyboard, **Gboard** [2]. It is used to improve **next-word prediction**, **query suggestion**, word completion, corrections [2], and **emoji prediction** [2]. FL is also adapted to learn **out-of-vocabulary (OOV) words** [2].
- **Mobile Operating Systems:** Used for features on **Pixel phones** and in **Android Messages** [1]. Apple uses cross-device FL in **iOS 13** for applications like the **QuickType keyboard** and the **vocal classifier for "Hey Siri"** [1].
- **General Mobile Apps:** Powers applications such as face detection and voice recognition [4]. Explored for **hotword detection** [1].
- **IoT Systems:** Used to limit vulnerabilities in large-scale IoT systems [2]. Use cases include **intrusion detection systems** based on anomaly detection [2] and **Mobile Edge Computing (MEC)** optimization [2].
- **Electric Vehicles (EVs):** Analyzing driver behavior metrics (using LSTM) to predict battery failure [2].

Organizational and Industrial (Cross-Silo FL)

- **Healthcare Informatics (HI):** FL is a strong fit for HI, balancing patient privacy with ML by keeping patient data on-premise [2]. Applications include:
  - Predicting mortality and hospital stay time using distributed **Electronic Health Records (EHRs)** [2].
  - Predicting hospitalizations for heart disease patients [2].
  - **Medical image prediction** (e.g., brain tumor segmentation) [2, **?**].
  - Collaborative learning for multi-modal **COVID-19 diagnosis** with X-ray and Ultrasound imagery [3].
  - Detecting **adverse events in mass-scale vaccination programs** [3].
  - Analysis of biomedical data (e.g., subcordinal brain changes in neurological disease) [2].
  - Drug discovery (e.g., MELLODDY project) [3].
  - Training on **multi-institutional datasets** constrained by HIPAA/FERPA [1].
- **Finance: Finance risk prediction for reinsurance** [1]. Collaborative training for **fraud detection** models [1]. Used by Webank and Swiss Re for high-precise financial analysis [3].
- **Recommender Systems:** Generating personalized recommendations using collaborative filtering [2].
- **Online Retailers:** Analyzing user click-stream data to enhance prediction of consumer's next browsing activities [2].
- **Model Types:** Applicable to models that operate over large inventories where clients interact with only a tiny fraction of items (sparse problems), such as **natural language models** or **content ranking models** using an embedding lookup table [1].

4 Advantages

FL provides significant advantages over traditional centralized and on-device machine learning models, primarily related to privacy, cost, and system efficiency.

- **Data Privacy and Security Guarantees:** FL is a **privacy-preserving decentralized approach** that keeps **raw data on-devices** and precludes direct access to it [2]. The approach involves sharing focused model updates (e.g., gradients) instead of the raw data [4]. Local training preserves the privacy, confidentiality, and integrity of patient data [3].
- **Compliance and Risk Mitigation:** FL can **mitigate many of the systemic privacy risks and costs** resulting from traditional centralized approaches [1]. It effectively handles the tradeoff between model learning and regulatory compliance (e.g., HIPAA/GDPR) because raw data is not centralized [3].
- **Communication and Network Efficiency:** FL eliminates the data communication overhead associated with centralized systems [2]. By pushing computation to the edge and requiring only small, iterative model updates, it reduces strain on the network [4].
- **Knowledge Sharing (vs. On-Device ML):** It overcomes the limitation of isolated on-device ML (where models don't benefit from peers' data) by aggregating local models to **share knowledge** among participants [2].
- **Accuracy and Model Diversity:** FL achieves **high precision and accuracy** by leveraging a large volume of data across many clients [2]. It also enables training on **multi-institutional datasets** (e.g., medical, education) where centralization was constrained, potentially leading to improved model **fairness** and diversity [1].
- **Resource and Latency Reduction:** FL handles the challenge of expensive centralized training [3]. It resolves the network latency problem as clients process data locally, eliminating the need to fetch data from a remote server [2].

5 Taxonomy of Attacks in Federated Learning

5.1 Overview

Federated Learning (FL) introduces distributed model training across multiple clients without centralizing data. While this paradigm preserves data locality, it exposes the system to unique security and privacy threats that differ from traditional centralized ML settings. These attacks can be categorized into five principal groups: Integrity, Privacy, Communication, Aggregator/Server, and Adversarial or Robustness attacks (adapted from Lyu et al., 2022; Nguyen et al., 2023).

Figure 4 provides an overview of the main attack surfaces in FL, grouped by layer: Integrity, Privacy, Communication, Aggregation, and Robustness.

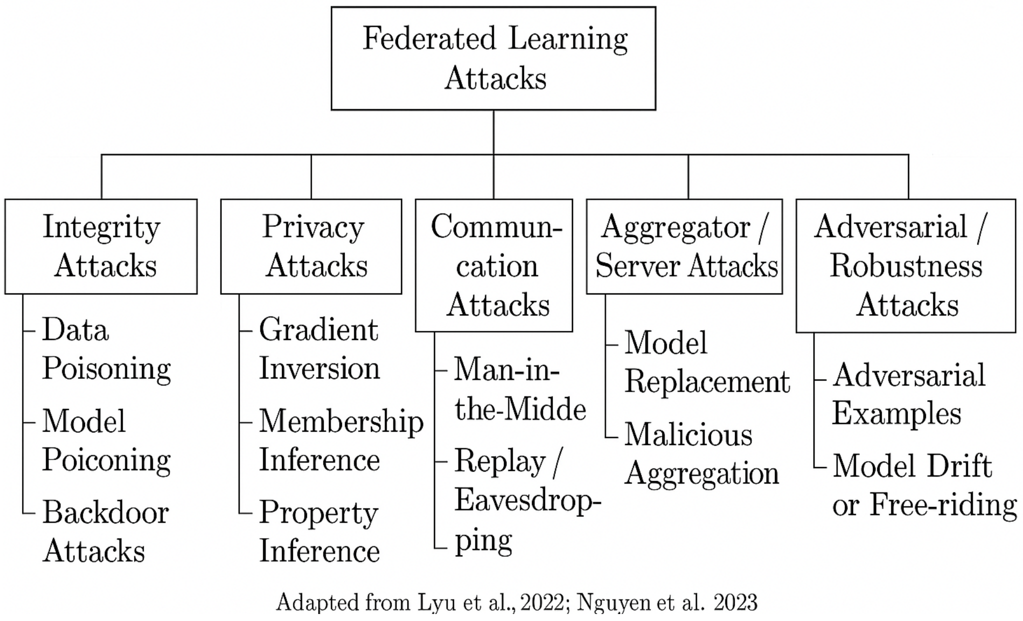

Figure 4: Taxonomy of Attacks and Defenses in Federated Learning

5.2 Integrity Attacks

Integrity attacks aim to compromise the correctness or reliability of the global model. Rather than stealing information, the adversary manipulates local updates to degrade model performance or introduce hidden behaviors. Typical examples include data poisoning—where malicious samples are injected into a client's dataset—and model poisoning, in which attackers directly modify the local model gradients before aggregation. Backdoor attacks constitute a more subtle variant: they embed a hidden trigger (e.g., a specific pixel pattern or phrase) that causes targeted misclassification while preserving overall accuracy. These threats are particularly difficult to detect in FL because of the asynchronous, privacy-preserving nature of local updates. (Bagdasaryan et al., 2020; Fung et al., 2020)

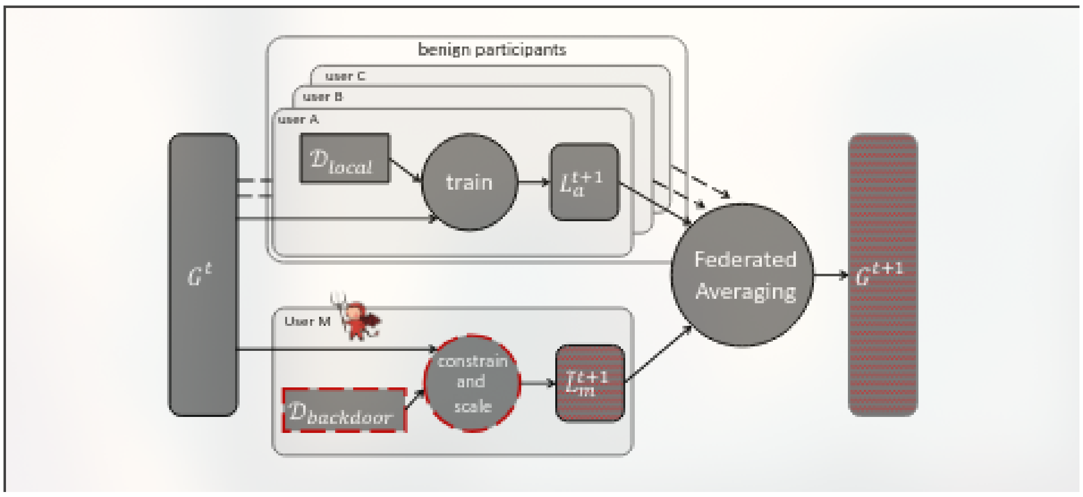

Figure 5: The attacker compromises one or more participants, trains on the backdoor data using the constrain-and-scale technique, and submits the resulting model, which replaces the joint model as the result of federated averaging. (Bagdasaryan et al., 2020; Fung et al., 2020)

5.2.1 Model poisoning

A compromised participant or a small group can directly modify the weights sent to the server so that the global model learns a hidden functionality (backdoor), while preserving performance on the main task.

- Unlike *data poisoning* where training examples are altered, the attacker manipulates the local model and the update that will be aggregated.
- The FL design (local updates, aggregation) gives participants direct influence and therefore an attack surface.

Main method — Model Replacement The attacker trains a malicious local model X that encodes the backdoor. They then construct the following submission:

$$\tilde{L} = \gamma \cdot (X - G_t) + G_t$$

- G_t: current global model.
- X: locally trained backdoor model.
- γ: scaling factor (chosen to compensate for dilution by aggregation of the average).

Logic: the aggregator computes a weighted average of the updates. For the malicious contribution to dominate the average, the attacker amplifies their update (γ) so that the resulting average is close to X. This enables *single-shot* injection in a single round if the update is large enough.
Limitations: a highly amplified update is theoretically detectable if the server observed individual contributions or their norms (but *Secure Aggregation* often prevents this visibility). Mechanisms such as *clipping* or *DP* can reduce the effect of large contributions.

Evasion: Constrain-and-Scale To evade defenses based on anomaly detection, the attacker does not simply apply a post-training scale. They train X by minimizing a multi-objective loss:

$$L = \alpha L_{class} + (1 - \alpha) L_{ano}$$

- L_{class}: the standard loss (main task + backdoor); trains the model to perform both the main task and the backdoor.
- L_{ano}: penalizes anomaly (e.g., high L_2 norm, low cosine similarity with benign update direction, drop in accuracy on an audit dataset).
- α: trade-off between strength and stealth.

Goal: produce individual updates that look like benign updates in magnitude and direction while enabling, once combined, the desired global effect.

Moment of injection

- Early attacks, before convergence, are more easily forgotten.
- Late attacks, near convergence, are much more persistent, because participants' updates become small and cancel out, leaving the backdoor intact.

Persistence and factors of effectiveness Persistence depends on: the rarity of the trigger in benign data, local/global learning rates, and the number of participants per round.

5.2.2 Distributed Backdoor Attacks (DBA)

DBA — *Distributed Backdoor Attacks.* Xie et al. present a backdoor attack specifically designed to exploit the distributed and heterogeneous nature of Federated Learning. Unlike classic attacks where the same trigger is injected locally by one (or several) client(s), DBA splits the global trigger into sub-triggers implanted by different clients. Each adversarial client introduces only a small part of the pattern; federated aggregation (FedAvg) then reassembles the backdoor at the global model level. This strategy makes the attack stealthier (local updates appear close to benign updates) and harder to detect by outlier-detection methods or by filtering large updates.

Differences with classic backdoor

- **Nature of the trigger:** classic backdoor — global trigger; DBA — distributed trigger, decomposed into several sub-patterns $\{\phi_i^*\}$ spread across multiple clients.
- **Strategy to dominate aggregation:** classic backdoor often uses model replacement (strong scaling of updates, large γ); DBA favors distributing the signal: each client sends only a small contribution.
- **Stealth / detection:** classic backdoor can be spotted if the aggregator monitors update magnitude or distance; DBA is better camouflaged because each individual update resembles a legitimate update.
- **Threat model required:** classic backdoor — one or a few compromised clients suffice; DBA — requires several compromised clients participating in targeted rounds.
- **Robustness to defenses:** classic backdoor can be mitigated by clipping, robust aggregation, or anomalous update detection; DBA circumvents several defenses designed for centralized backdoors.

Mathematical formalism

$$\min_w F(w) = \frac{1}{N}\sum_{i=1}^{N} f_i(w)$$

$$G^{t+1} = G^t + \eta\frac{1}{n}\sum_{i=1}^{n}(L_i^{t+1} - G^t)$$

$$L_i^{t+1} \leftarrow \gamma\,(w_i - G^t) + G^t \quad (\gamma large)$$

$$w_i^* = \arg\max_{w_i}\ \Big(\sum_{j\in S_{poi}^i} \Pr\left[G^{t+1}(R(x_j^i, \phi_i^*)) = \tau\right] + \sum_{j\in S_{cln}^i} \Pr\left[G^{t+1}(x_j^i) = y_j^i\right]\Big)$$

DBA introduces the decomposition $\phi \to \{\phi_i^*\}$: the ϕ_i^* are designed to be weak locally but cooperate to produce a global trigger after aggregation.

5.2.3 Sybil attacks

A **sybil** is a fake or multiple client identity controlled by the same attacker. Objective: amplify the malicious influence on the global aggregate by multiplying contributions.

Types of Sybil-related attacks

- **Model poisoning**: sending malicious gradients to bias the global model.
- **Label-flipping**: flipping local labels.
- **Backdoor**: injecting a discrete trigger causing misclassification when the trigger is present.

Why it is dangerous

- A single real attacker can spawn n sybils and multiply their influence.

5.3 Privacy Attacks

Privacy attacks focus on recovering or inferring information about clients' training data from shared model parameters or gradients. Despite not having access to raw data, attackers can exploit mathematical properties of the gradients to reconstruct private inputs (gradient inversion, Zhu et al., 2019) or determine whether a specific data point was used in training (membership inference). More sophisticated approaches—such as property inference—can deduce aggregate demographic or categorical information about local datasets. These attacks demonstrate that FL alone does not guarantee privacy; without cryptographic protection or differential privacy, model updates remain vulnerable to inversion and inference analysis. (Melis et al., 2019)

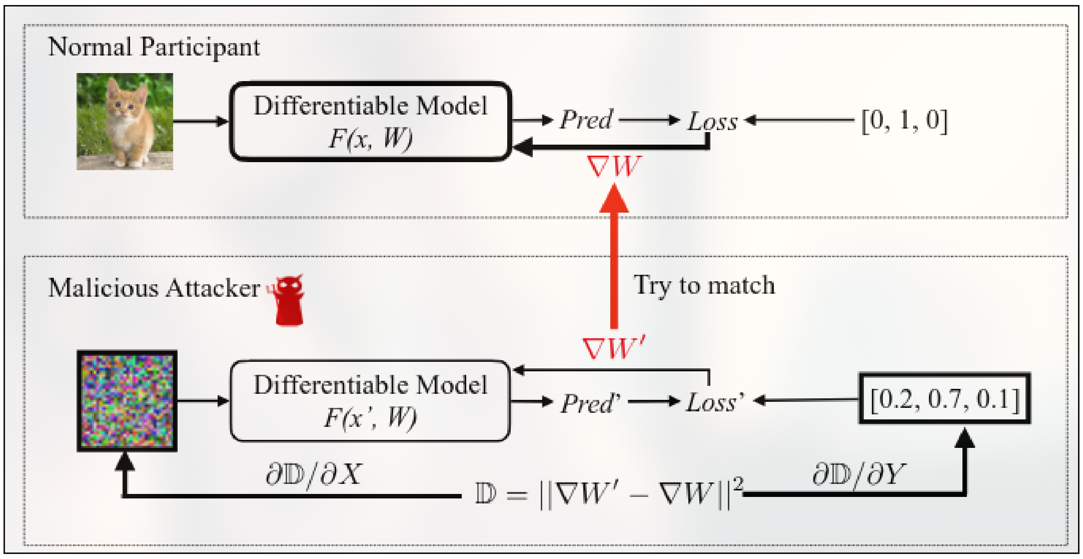

Figure 6: The overview of the DLG algorithm. Variables to be updated are marked with a bold border. While normal participants calculate deltaW to update parameter using its private training data, the malicious attacker updates its dummy inputs and labels to minimize the gradients distance. When the optimization finishes, the evil user is able to steal the training data from honest participants. (Zhu et al. 2019)

5.4 Communication Attacks

In communication attacks, the adversary targets the exchange channel between clients and the central server. Since FL involves frequent transmission of model updates, a man-in-the-middle or replay attacker can intercept, modify, or resend parameter messages. Eavesdropping on updates may reveal sensitive information about gradients or hyperparameters. Securing FL communication requires authenticated and encrypted channels, but even then, timing or metadata side-channels may leak information (Lyu et al., 2022).

5.5 Aggregator / Server Attacks

In a standard FL setting, the server is assumed to be semi-honest—but this assumption may not hold in adversarial contexts. A malicious or compromised server can perform model replacement (substituting the aggregated model with a poisoned version) or malicious aggregation (selectively weighting updates to bias outcomes). Conversely, even honest servers can unintentionally enable these attacks if they rely on naive averaging (FedAvg) without anomaly detection or robust aggregation. Hence, secure aggregation protocols and verifiable model updates are crucial to maintaining integrity at the central level. (Bhagoji et al., 2019)

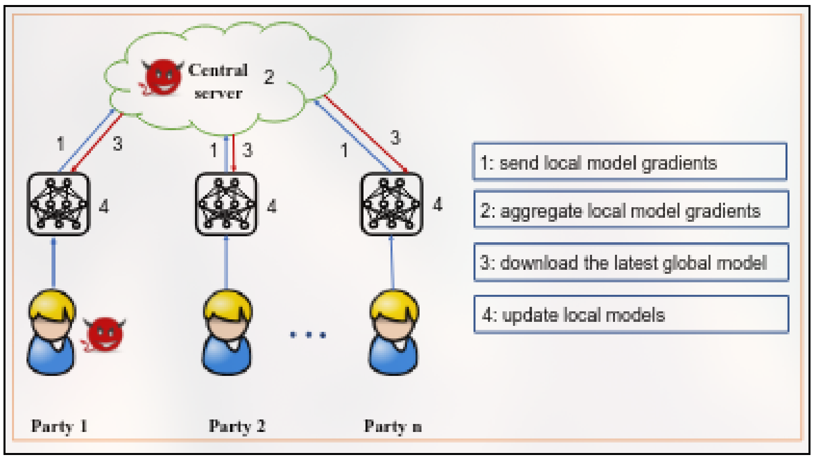
Central server
2
1
3
1
3
1
3
4
4
4
1: send local model gradients
2: aggregate local model gradients
3: download the latest global model
4: update local models
Party 1
Party 2
Party n

Figure 7: A typical FL training process, in which both the (potentially malicious) FL server/aggregator and malicious participants may compromise the FL system. (Threats to Federated Learning: A Survey, Lyu et al.)

5.6 Adversarial / Robustness Attacks

Beyond the training process itself, adversarial or robustness attacks target the learned model's behavior during inference. Attackers craft adversarial examples—inputs with imperceptible perturbations that cause misclassification or exploit model drift and free-riding phenomena, where participants benefit from others' updates without contributing valid gradients. These attacks highlight that FL systems must combine robustness evaluation, adversarial training, and continuous monitoring to maintain resilience against both internal and external threats. (Nguyen et al., 2023)

Table 1: Comprehensive Mapping of Federated Learning Attacks

Category	Subtype	Description	Key References
Poisoning Attacks	Data Poisoning	Manipulation of local datasets to bias global model updates	[5, 6, 7]
	Model Poisoning	Direct modification of gradients or model weights to introduce hidden behavior or degradation	[8, 9]
Backdoor Attacks	Trigger-based Backdoors	Introduction of hidden malicious patterns (triggers) that cause misclassification when activated	[5]
Inference Attacks	Membership Inference	Determining whether specific data points were used in training	[10, 11]
	Property Inference	Inferring sensitive attributes of users' data from gradients or model parameters	[11]
	Gradient Inversion	Reconstructing training samples from shared gradients	[12, 13]
Communication Attacks	Man-in-the-Middle (MITM)	Intercepting or modifying updates in transit between clients and server	[14]
	Sybil Attacks	A single adversary simulates multiple fake clients to bias aggregation	[15]
Free-rider Attacks	Model Theft	Participants submit fake updates but still benefit from global model	[16]
Model Replacement	Model Overwrite	Substituting the global model with a maliciously crafted one	[5]

5.7 Representative Works

Table 2 summarizes key representative works, their publication venues, and their classification in the CORE database.

Attack Type	**Representative Paper**	**Venue**	**CORE Rank**	**Description**
Data Poisoning	Bagdasaryan et al., 2020	AISTATS	A	Demonstrates model-replacement and backdoor attacks targeting the global model aggregation in FL.
Gradient Leakage / Inference	Zhu et al., 2019	NeurIPS	A*	Shows that private training data can be reconstructed from shared gradients.
Property / Membership Inference	Melis et al., 2019	IEEE S&P	A*	Reveals feature leakage through model updates during collaborative training.
Privacy Analysis	Nasr et al., 2019	IEEE S&P	A*	Formalizes white-box inference attacks against centralized and federated models.
Byzantine / Robustness	Blanchard et al., 2017	NeurIPS	A*	Introduces Byzantine-tolerant gradient aggregation methods for adversarial clients.
Sybil / Collusion Attacks	Fung et al., 2020	RAID	A	Demonstrates Sybil-based poisoning and proposes FoolsGold defense.
Distributed Backdoors	Xie et al., 2020	ICLR	A*	Introduces DBA attacks — coordinated small-scale poisoning to evade detection.
GAN-based Reconstruction	Hitaj et al., 2017	ACM CCS	A*	Early work using GANs to extract data from collaborative learning setups.

Table 2: Mapping of Attacks in Federated Learning: representative works and venues.

6 Defense Mechanisms in Federated Learning

Federated Learning defenses aim to mitigate threats without violating its core principle of decentralized data ownership. Defense research has evolved from classical Byzantine-tolerant learning toward integrated privacy-preserving and robustness-preserving frameworks. This section synthesizes the state-of-the-art (SOTA) mechanisms for each attack category—Integrity, Privacy, Communication, and Aggregator/Robustness—emphasizing their underlying principles, current limitations, and research gaps.

6.1 Integrity: Data and Model Poisoning, Backdoor Attacks

Foundations. Early Byzantine-robust methods focus on filtering anomalous client updates before aggregation. *Krum* and *Multi-Krum* select updates closest to the majority in Euclidean space, tolerating up to a fixed fraction of malicious clients [7]. *Trimmed Mean* and *Median* aggregation perform coordinate-wise pruning of extremes, while *Bulyan* improves stability by combining both selection and trimming phases [17, 18]. These methods remain core baselines for integrity, but they assume i.i.d. data and synchronous participation—assumptions rarely valid in modern FL deployments.

Robust aggregation: Recent approaches adapt robustness to non-i.i.d. and large-scale FL. **FLTrust** [19] introduces a trusted reference dataset on the server, computes a reference update, and scores each client update by cosine similarity to this reference. Updates are normalized to prevent large outliers, then weighted by their trust scores. FLTrust achieves strong resilience against data and model poisoning—even with over 40% malicious clients—but depends on server-side clean data. Complementary work introduces adaptive robust aggregation (e.g., momentum-based filtering or clustering-driven aggregation) that identifies groups of coherent updates rather than outlier removal alone.

In parallel, efficient statistical defenses such as the *Coordinate-wise Median of Means* and *Geometric Median* reduce computational overhead while preserving bounded-influence properties. Recent quasi-linear-time robust mean estimators further close the gap between robustness and scalability.

Anomaly and outlier detection. Beyond rule-based aggregation, anomaly detection employs machine learning itself. Autoencoder-based detection and spectral clustering approaches learn the manifold of normal updates and isolate deviations. Some frameworks (e.g., AlignIns, Xu et al., 2025) analyze directional alignment of gradient vectors to capture stealthy manipulations that mimic benign norms but differ in orientation. Dual-layer filtering first partitions updates by local density (trust vs. suspicious groups), then re-evaluates residuals in each group, improving recall without sacrificing fairness to heterogeneous clients. These detection systems, however, require careful threshold tuning and historical context to distinguish diversity from malicious deviation.

Backdoor-specific defenses. Backdoor attacks [5, 20, 16] exploit the fact that the global model generalizes triggers seen by only a few clients. SOTA defenses combine several complementary strategies:

- **Norm clipping:** limits each update's ℓ_2 norm, mitigating single-round model replacement. Adaptive attackers may, however, scale updates just below clipping thresholds.
- **Directional filtering:** compares each update's cosine similarity with both the global and majority directions to suppress deviating gradients.
- **Server-side audits:** evaluate updated models on a clean validation set or known canary triggers to detect accuracy drops or suspicious activations.
- **Model repair:** post-hoc pruning or fine-tuning on clean data, removing neurons strongly correlated with triggers (akin to Neural Cleanse).

DBA-style distributed triggers remain challenging because each adversarial client introduces only a fragment of the backdoor. Detecting these requires multi-round consistency checks or frequency analysis over update histories.

Trade-offs and insights. Robust aggregation ensures integrity but can degrade accuracy under data heterogeneity. Many algorithms assume bounded malicious fractions (often below 25–40%) and synchronous updates. Anomaly detection improves flexibility but introduces computational overhead and potential false positives. Privacy measures like *Secure Aggregation* conceal individual updates, making robust filtering harder—highlighting a persistent privacy–robustness tension.

6.2 Privacy and Inference: Gradient Inversion, Membership, and Property Attacks

Differential Privacy (DP). Differential privacy has become the dominant defense paradigm for inference risks. FL implementations typically use *DP-FedAvg* or *DP-FedSGD*, where each client clips gradients and adds Gaussian noise before sending updates. The added noise guarantees (ε, δ)-DP, bounding how much any individual's data can influence the global model [12, 13, 10, 11]. Modern DP research improves the privacy–utility balance via:

- Adaptive noise scheduling (less noise in later rounds as gradients stabilize),
- Layer-wise noise scaling—allocating privacy budgets per model layer,
- Advanced privacy accounting (e.g., Rényi DP or zCDP) for tighter bounds.

While DP thwarts direct gradient inversion, it can slow convergence and disproportionately impact small or non-i.i.d. clients.

Cryptographic defenses. Secure Aggregation (*SecAgg*) [21] ensures the server learns only the aggregated sum of client updates using additively homomorphic encryption and secret sharing. Even a curious server cannot inspect individual gradients, significantly reducing gradient leakage risk. SOTA variants (e.g., SecAgg+) enhance dropout tolerance and reduce bandwidth overhead for thousands of clients. Homomorphic encryption (HE) and secure multi-party computation (MPC) protocols extend this idea—allowing operations directly on encrypted updates—but remain resource-heavy. Systems such as FedML-HE [22] and hybrid approaches (HE + DP) balance confidentiality with performance, albeit at added complexity.

Gradient obfuscation and compression. Practical obfuscation (e.g., sparsification, quantization, or random rotation of gradients) limits information content. Although not providing formal DP guarantees, such schemes complicate inversion attacks and improve communication efficiency. Combined with DP or SecAgg, they provide layered protection with minor utility cost.

Trade-offs. DP offers formal privacy but at a measurable cost to accuracy. Strong noise (small ε) hinders utility, while weak noise leaves leakage. SecAgg adds communication rounds and depends on synchrony among clients. Combining DP and cryptography—common in production FL—offers strong guarantees but increases latency and server complexity. Importantly, privacy defenses often reduce visibility for anomaly or Sybil detection, requiring new privacy-compatible audit mechanisms.

6.3 Communication and Sybil Defenses

MITM and transport security. Communication-layer attacks are addressed with standard cryptographic protocols. FL frameworks (e.g., TensorFlow Federated, PySyft) integrate TLS/SSL encryption and mutual authentication. Model updates can be digitally signed or accompanied by message authentication codes to ensure integrity. While such measures prevent classic man-in-the-middle tampering, they do not protect against compromised endpoints or malicious clients—highlighting the need for higher-layer defenses [14].

Sybil detection and mitigation. Sybil attacks—where an adversary spawns multiple fake clients—amplify poisoning power. Algorithmic defenses such as **FoolsGold** [15] detect Sybils via update similarity: clients exhibiting consistently high cosine similarity across rounds receive reduced learning rates. FoolsGold adapts weighting dynamically, suppressing clusters of correlated updates without requiring explicit identification. Follow-up works explore:

- Clustering-based defenses that identify communities of similar updates;
- Reputation and staking mechanisms penalizing unhelpful or redundant clients;
- Randomized client sampling per round to probabilistically limit Sybil influence.

Nonetheless, adaptive Sybils can introduce noise to decorrelate gradients, evading similarity checks. Moreover, highly correlated honest clients (e.g., from similar data distributions) may be unfairly down-weighted, underscoring fairness–security trade-offs.

6.4 Aggregator and Robustness: Model Replacement, Free-Riding, Adversarial Examples

Model replacement. To prevent malicious model substitution [5, 16], servers enforce per-update clipping, employ robust aggregation (median, Krum, FLTrust), and maintain validation-based audits to detect abrupt shifts in accuracy or decision boundaries. Advanced strategies compute the cosine divergence between consecutive global models or employ checksum-based verification in distributed aggregation (e.g., blockchain consensus). While effective, such redundancy increases training latency and requires careful fault tolerance.

Free-rider detection. Free-riders exploit FL by submitting trivial or fabricated updates. Techniques like FRIDA (Recasens et al., 2024) apply inference-based audits: if a client's claimed dataset leaves no detectable trace in the updated model (e.g., via membership inference), the client is flagged as non-contributing. Simpler heuristics—tracking gradient norms, parameter change frequency, or local validation accuracy—also provide signals. However, DP or SecAgg can obscure such metrics, necessitating privacy-preserving contribution scoring or cryptographic proofs of work.

Adversarial robustness. Adversarial examples threaten inference integrity. **Federated Adversarial Training (FAT)** distributes adversarial training to clients: each generates perturbed samples (e.g., FGSM or PGD attacks) and optimizes local robustness objectives. Calibrated FAT (CalFAT) [23] addresses label skew by aligning local logits, ensuring stable aggregation across diverse distributions. Empirically, FAT improves robustness to L_∞-bounded perturbations but at substantial computational cost and reduced clean accuracy. Hybrid methods combine centralized adversarial fine-tuning with federated updates to balance cost and robustness.

Trade-offs. Defenses at the aggregator layer balance correctness, scalability, and transparency. Strong clipping and aggregation slow convergence; decentralized validation or blockchain consensus adds latency. FAT improves robustness but increases local computation and bandwidth. Integrating these defenses under constrained devices remains a core engineering challenge.

Conclusion

Federated Learning represents a paradigm shift in machine learning—one that enables large-scale collaboration without centralizing sensitive data. Yet, this decentralization also expands the attack surface, exposing new vulnerabilities across data, model, communication, and aggregation layers. This state-of-the-art survey has demonstrated that while FL offers strong privacy-by-design principles, its inherent openness requires a continuous, multi-layered defense strategy to ensure trustworthiness and operational resilience.

Robust aggregation mechanisms such as Krum, Bulyan, and FLTrust have matured to defend against poisoning and backdoor attacks, while privacy-preserving frameworks based on Differential Privacy and Secure Aggregation address gradient leakage and inference threats. Cryptographic protocols, anomaly detection, and adversarial training extend this protection to cover communication integrity and robustness against adaptive adversaries. However, each defense introduces trade-offs between privacy, accuracy, scalability, and interpretability, underscoring the need for holistic system design.

Traditional machine learning pipelines must evolve into secure, self-monitoring ecosystems capable of anticipating, detecting, and mitigating attacks in real time. Future MLSecOps pipelines for FL will integrate continuous monitoring, automated patching, and dynamic defense orchestration while maintaining compliance with data governance regulations.

Ultimately, the path forward lies in harmonizing security, privacy, and performance through layered, interoperable defenses. By embedding security as a priority throughout the FL lifecycle—from client selection to model deployment—Federated Learning can achieve both the scalability demanded by modern AI systems and the resilience required for real-world, adversarial environments.

References

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, 2019.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji *et al.*, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Transactions on Machine Learning and Knowledge Extraction*, 2021.

[3] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas *et al.*, "Adoption of federated learning for healthcare informatics: Emerging applications and future directions," *npj Digital Medicine*, 2020.

[4] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, 2020.

[5] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "Backdoor attacks on federated learning," *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[6] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," *USENIX Security Symposium*, 2020.

[7] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[8] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[9] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[10] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[11] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[12] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[13] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[14] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *ACM Computing Surveys*, 2020.

[15] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *Proceedings of the 21st International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2018.

[16] Y. Wang, Y. Hu, G. Li, and Z. Lin, "Attack of the tails: Yes, you really can backdoor federated learning," *USENIX Security Symposium*, 2021.

[17] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Machine Learning (ICML)*, 2018. [Online]. Available: https://arxiv.org/abs/1803.01498

[18] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *Proc. Int. Conf. Machine Learning (ICML)*, 2018. [Online]. Available: https://arxiv.org/abs/1802.07927

[19] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *Proc. Network and Distributed System Security Symposium (NDSS)*, 2021. [Online]. Available: https://arxiv.org/abs/2012.13995

[20] C. Xie, K. Huang, P. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2020. [Online]. Available: https://openreview.net/forum?id=rkgyS0VFvr

[21] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for privacy-preserving machine learning,” in *Proc. ACM Conf. Computer and Communications Security (CCS) Workshop*, 2017. [Online]. Available: https://arxiv.org/abs/1705.10816

[22] Y. Liu, J. Zhang, Y. Zhang, Z. Lin, and J. Huang, “Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system,” *IEEE Transactions on Information Forensics and Security*, 2021.

[23] C. Chen, Y. Liu, X. Ma, and L. Lyu, “Calfat: Calibrated federated adversarial training with label skewness,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. [Online]. Available: https://openreview.net/forum?id=H8wCqfY3h8