ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Scuola di Ingegneria e Architettura
Dipartimento di Informatica, Scienza e Ingegneria · DISI
Corso di Laurea Magistrale in Ingegneria Informatica

PYOD: Pwn Your Own Device

Presented by:
Riccardo Bovinelli
Luca Cimino
Lorenzo Riccardi
Tommaso Sgreccia

Project Presentation for:
Cybersecurity M

Introduction to Federated Learning

Limitations in Centralized Machine Learning

- **Machine Learning**: develop models that learn from training data to perform predictions on unseen test data
    - Typically, data are generated by heterogeneous devices (e.g. IoT sensors)
    - *Classical approach*: data collection and training phase are done in a central server

- **Limitations** in the centralized approach:
    - *Regulations*: different data protection regulations across the world
    - *User Privacy*: users do not want their private data to leave their device
    - *Scalability*: high efficiency requirements for raw data collection

Introducing Federated Learning

- **Federated Learning**: enables cooperative ML by moving the training phase to each participant device
  - ***Main assumption***: private data is maintained local to each participant (client)
  - A centralized component (server) aggregates local model parameters into a global model

- Many different **privacy-enhancing applications**:
  - Financial fraud detection by different organization's data collection
  - Sharing healthcare records for various disease detection
  - Intra and inter-organization use of industrial IoT devices data
  - Privacy preserving model training for IDS across multiple networks
  - …

Federated Learning Workflow

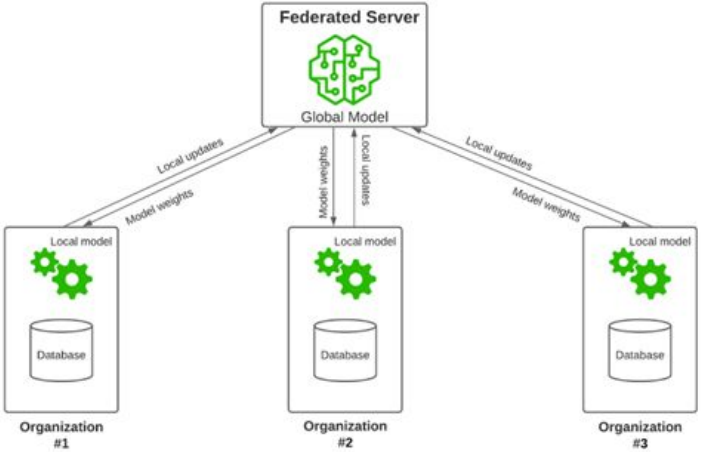
1. **Initialization of the model** parameters on the server and sending on client nodes

2. **Local Training Phase** on each client node

3. **Sending Local Parameters** to the server

4. **Aggregation** of collected Model Parameters

5. Repeat until model converges



Drawbacks of Federated Learning

1. **Larger attack surface**: both the server, and now also the clients, can be compromised
2. **Communication overhead** between clients and server: each round is composed by two data exchanges per client
3. **Unreliability** of the devices that contribute to the system: disconnections and (potentially) lost updates are sometimes critical
4. Still, **privacy concerns**: we will see them soon

Threats and Vulnerabilities in FL

Threat Model of Federated Learning

➔ Privacy and Security Issues

- *Priorities*: privacy and data leakages ↔ attacks to the system security

➔ Insider and Outsider Threats

- *Outsider threats* → eavesdropping over communication channel, MiTM, etc.
- *Insider Threats* → compromised participants launch attacks to the system
  - Higher risks: model tampering, privacy leakage, etc.

➔ Threats at Training and Inference Phase

- *Training-time threats*: compromise the integrity of dataset or local models
- *Inference-time threats*: collect information about models to extract private data

Vulnerabilities of Federated Learning

➔ Communication channels

- Several training rounds between multiple clients required for convergence
  - Insecure channel → *open vulnerability*
- Communication bottlenecks and failures discard clients → weakened model

➔ Clients and Data Reconstruction

- *Clients*: access to intermediate model snapshots and training updates at each round
  - *Malicious Clients*: tamper the training process or infer private data

- Attackers exploit model updates to compute gradients and reconstruct private data

Vulnerabilities of Federated Learning (cont.)

➔ Malicious Server and Aggregation Algorithm

- Aggregation Algorithm: should incorporate mechanisms to detect abnormal updates
- *Server*: initialize and share the global model, aggregate updates
  - *Malicious Server*: compromise aggregation phase and inspect private client updates

➔ Model Deployment

- Focus on FL robustness to adversarial attacks incorporated during training
  $\rightarrow$ *vulnerable deployed model*
- Robustness both at training time and at inference time

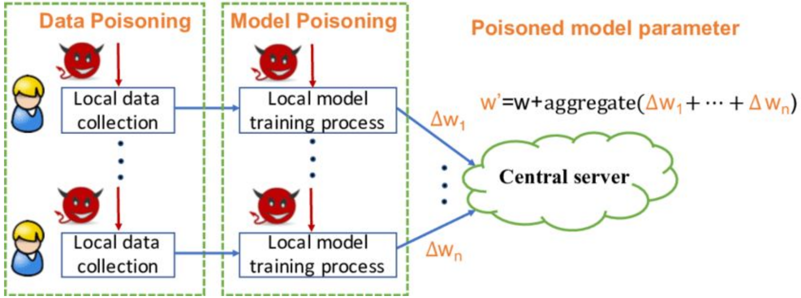
Taxonomy of Attacks in FL

Attacks exploiting data

- **Poisoning attacks**: when the client, at training phase, tamper either the local training set or the local model training procedure
  - *Data poisoning*
  - *Model poisoning*
- **Backdoor attacks**: a particular type of model poisoning that aims to inject a backdoor in the model to trigger misclassifications under certain input patterns.
- **Evasion attacks**: during testing phase some testing inputs are purposely misclassified, aiming at misprediction on specific inputs

Poisoning attacks

- **Data poisoning**: generation of dirty samples to tamper the model behaviour
  - **Dirty label**: the attacker can access and alter labels of any training data (e.g. *label flipping*)
  - **Clean label**: the opposite, but considering the attacker can replicate the data he wants to misclassify and set them the desired labels
- **Model poisoning**: here, we aim to tamper the model parameters directly before sending them to the server

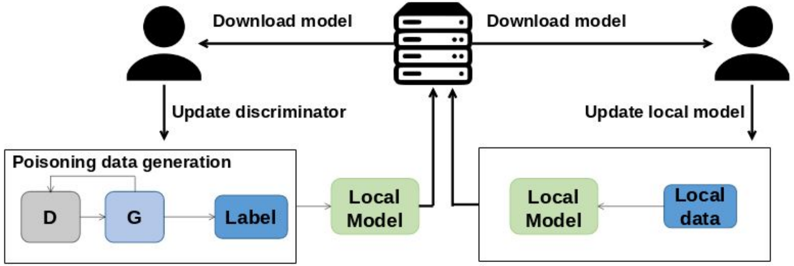


Attacks exploiting the Federation

- **Inference attacks**: the goal of the attacker is to obtain information about the other clients and the original training set
- **Free-riding attacks**: passive attack where the attacker is interested in obtaining the working model without actively participating in the iterative model update.

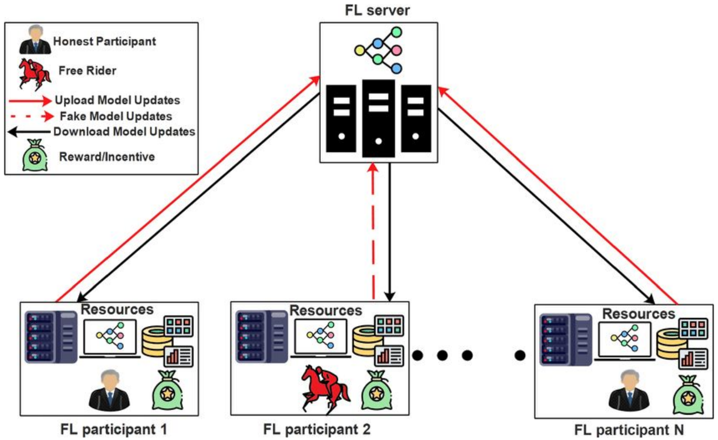
Inference attacks

- Apart from *gradient leakage* (which can expose, indirectly, personal training data and therefore users' private informations), it has been shown an attacker can recover the initial training set without any knowledge (**inference attack**)
  - **Membership inference attack**
- **Generative Adversarial Networks** (*GANs*) provide an efficient way for an attacker to achieve these results



Free-riding attacks

- Among the various attacks examined, this is the only one which does not result in a degradation of the global model
- Knowledge is achieved by global model updates
- The attacker sends *fake* updates (resulting in a “**free-ride**”) to the server
- The server replies, as by protocol, with the actual updates of the global model



Our implementation

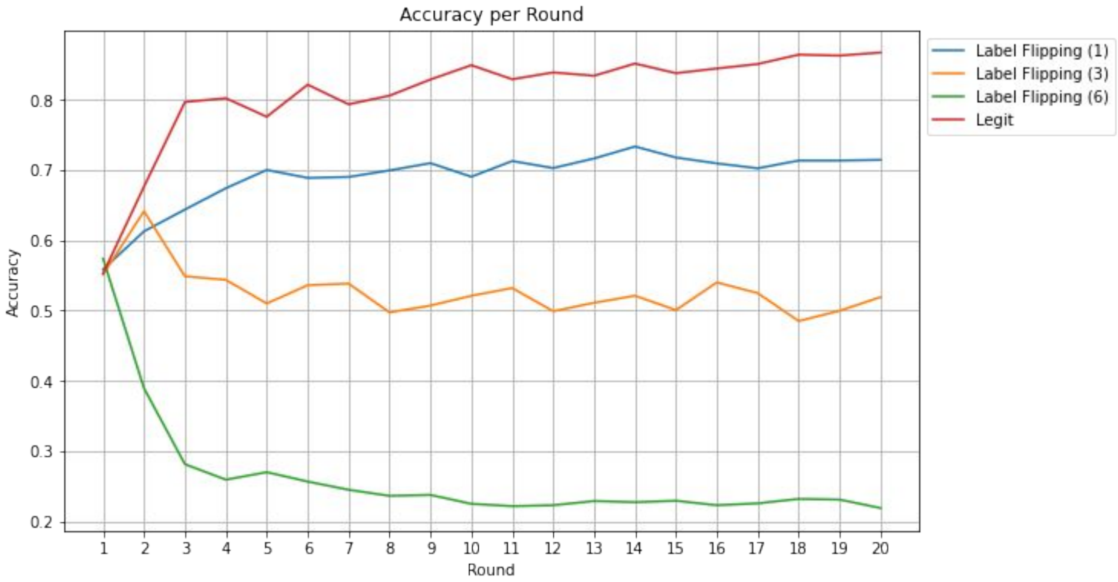
Label flipping attack - Overview

➔ **Goal of the attacker**: mislead the aggregated global model by deliberately changing the labels of their local training data

➔ <u>**Algorithm**</u>: Given a training dataset with training labels y_{train}

1. $y_{flip} = y_{train}$
2. **for each** label y_{flip} **do**
3. swap $y_{flip}[1]$ with $y_{flip}[2]$
4. $y_{train} = y_{flip}$
5. **end for**

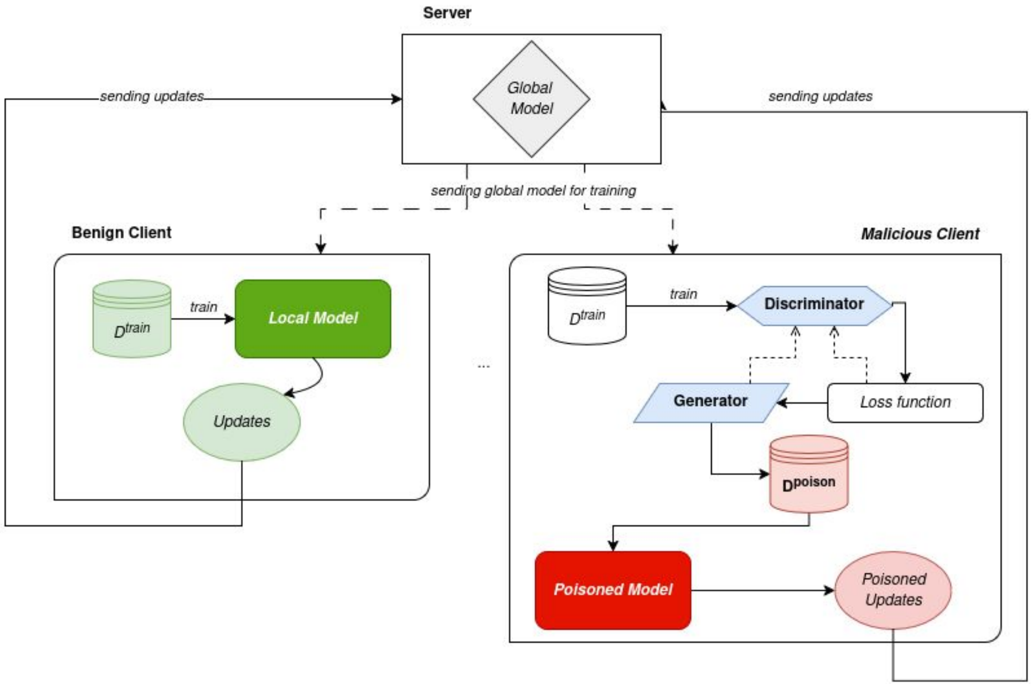
Label flipping attack - Results



➔ As expected, the more adversarial clients we have, the more the aggregated accuracy decrease

GAN-based model poisoning attack

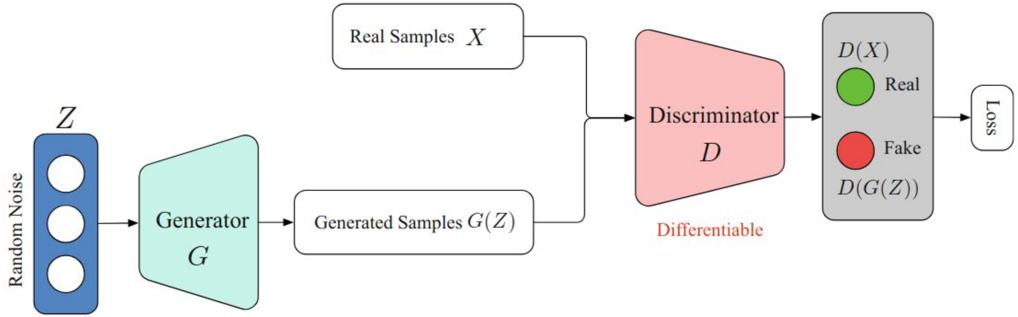
➔ **Goal of the attacker**: generate a fake malicious dataset that will produce poisonous local model updates, degrading the overall performances.



GAN-based model poisoning attack - Algorithm

With a Generator (**G**) and a Discriminator (**D**), the Attacker wants to create fake instances of class.
For each round, at GAN training:

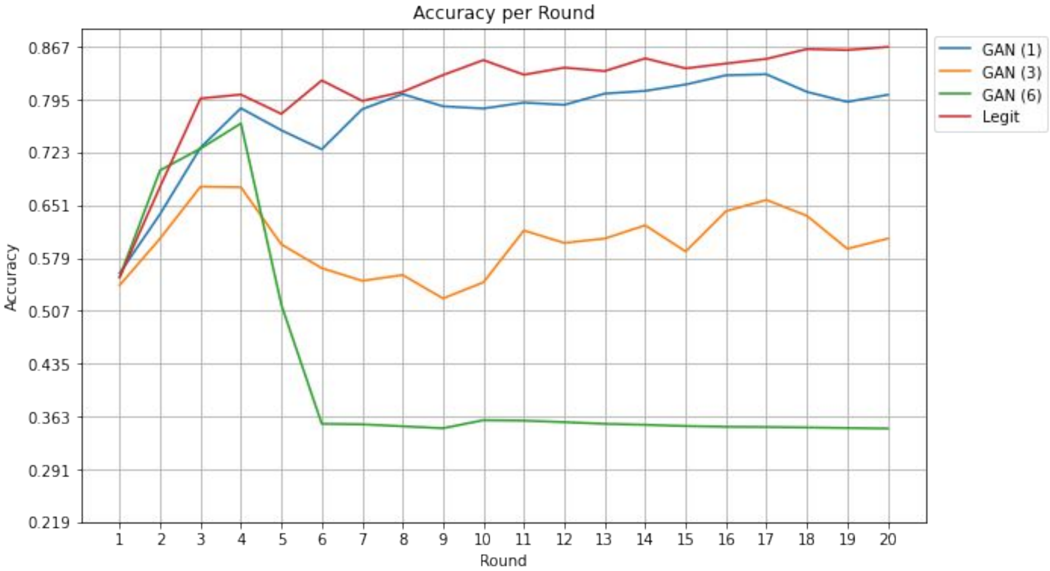
1. Train D on the real D^{train}
2. We generate poisoned data from G
3. Compute the loss function for G and D, and update their parameters
4. We include the new poisoned data batch in the poisoned dataset D^{poison}



Then, at model training, at each round:

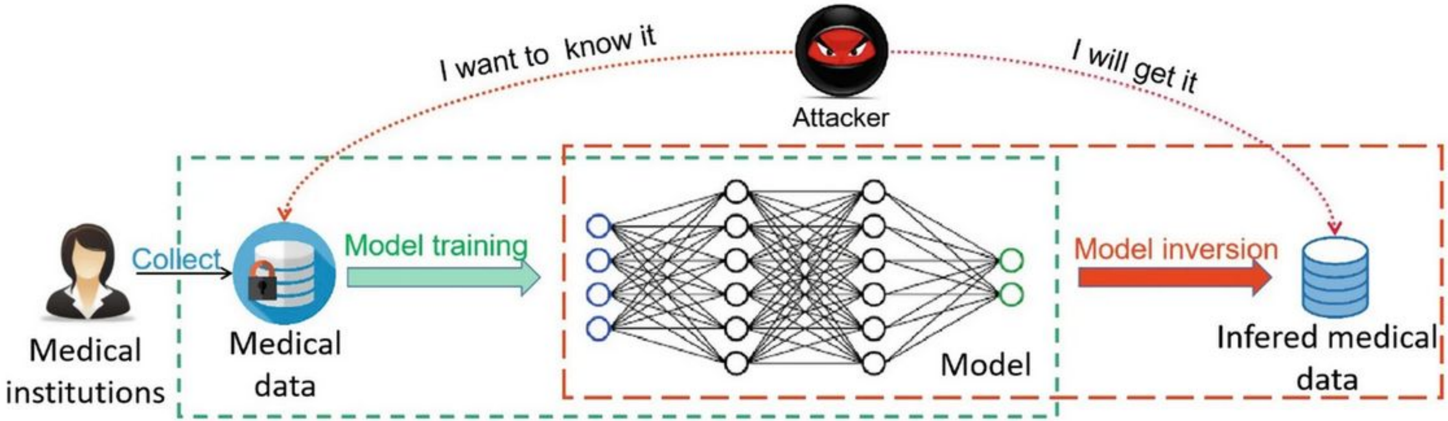
1. Update the local model from the global model
2. Train the model with the poisoned dataset
3. Compute the loss function and generate the weights
4. Update the new gradients to the server

GAN-based model poisoning attack



➔ Even there, as expected, the more adversarial clients we have, the more the aggregated accuracy decrease

GAN-based inference attack



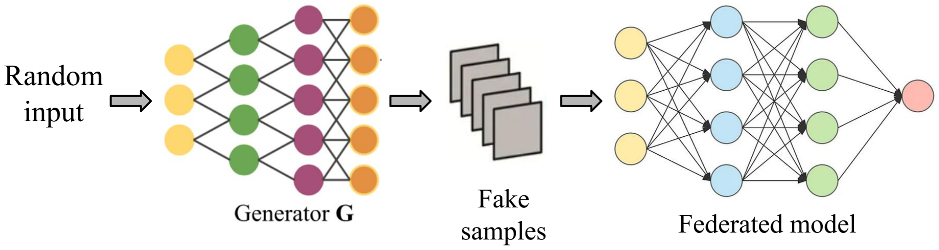
➔ **Goal of the attacker**: generate samples that appear to come from the victim's private training set, without having access to them.

GAN-based inference attack - Algorithm

With a Generator (**G**), the Attacker wants to create fake instances of class **C**. For each round:

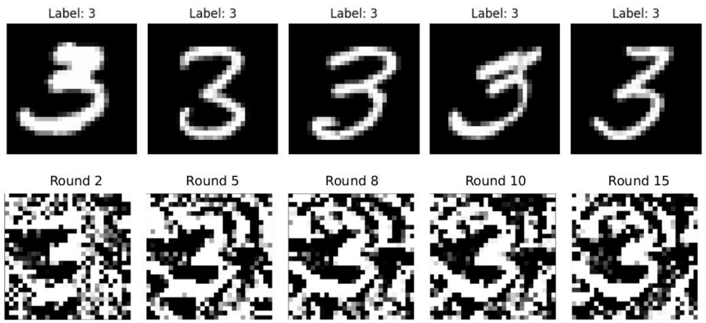
1. Update the local model with the global one
2. Generate fake samples with G
3. Use the local model to compute the probability of each fake sample to be of class C
4. Compute the loss function for G on the predicted probability
5. Mislabel all the samples generated by G
6. Train the local model with those samples
7. Upload the parameters to the server

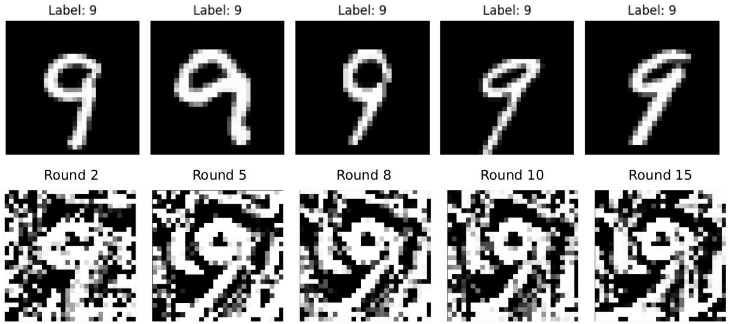
The **attacker** combines a **Generator network** with the federated model



GAN-based inference attack - Results

➔ Tests with MNIST dataset





- **Effect**: the attacker can extract relevant information about a class that was supposed to be private
- **Limitation**: unsuitable for situations where the goal is to target a specific client rather than an entire class of data

Defending from FL attacks

Defense in a Federated Learning context (cont.)

- **Anomaly detection and robust aggregation**
  - **Proactive** measure: detection of malicious updates
  - Achieved through classification methods: e.g. *Krum* or *AUROR*
  - A **client reputation score** can also be employed
- **Pruning and Fine-Pruning**
  - *Goal*: **remove** ("prune") *dormant* neurons, activated only during a backdoor attack
  - *Fine-Pruning*: local retraining phase on the server against pruning-aware attacks
- **Adversarial Training**
  - If you cannot avoid dirty updates from byzantine clients, *prepare* for them by training with fake data along benign one

Defense in a Federated Learning context (cont.)

➔ Privacy Protection Techniques

- **Secure Multi-Party Computation**: inference attacks mitigation, and should be a standard measure for each FL deployment
- **Homomorphic Encryption**: operations are performed directly over the cyphertext without prior decryption, keeping final results encrypted
  - *Partial HE*: one-way, one-time operation on encrypted data
  - *Full HE*: full support for bidirectional and multiple transformations on cyphertext
- **Differential Privacy**: introducing noise to client updates, avoiding inference
- **Trusted Execution Environment**: isolating aggregation and gradient computation environments improves **privacy** for each party involved

Against our attacks…

- ➔ **Label-flipping** and **GAN-based Model Poisoning**
  - ◆ *Anomaly Detection and Robust Aggregation*: by discarding model updates that are significantly different from the others during aggregation, we can achieve good results defending from LF
  - ◆ *Adversarial Training*
- ➔ **GAN-based Inference attack**
  - ◆ *Adversarial Training* (*Anti-GAN mechanism*): by training both on client's private dataset and on a fake dataset, and then using them mixed to perform the training.
    - Training X = Real X + Fake X'
  - ◆ *Differential Privacy*: almost no improvements, and moreover there is a significant degradation of the overall accuracy in the global model

Thank you for your attention!