Project Group Work in Cybersecurity M

PYOD: Pwn Your Own Device

Spezia Nicolò, Di Giovanni Roberto, De Simoni Clarissa

Anno Accademico 2024/2025

Sommario

Abstract

Il Federated Learning (FL) rappresenta una metodologia emergente di Machine Learning distribuito, caratterizzata da una crescente diffusione e da una serie di vulnerabilità intrinseche. Questo report ha un duplice obiettivo: da un lato, analizzare le strategie di attacco più avanzate e rappresentative nello stato dell'arte del FL, evidenziando le principali vulnerabilità e i limiti delle difese attuali; dall'altro, implementare e valutare degli attacchi specifici, misurandone l'impatto sulle prestazioni del modello e confrontandolo con uno scenario privo di minacce.
In quest'ultima sezione, presentiamo la descrizione, i risultati ottenuti e le prestazioni dei seguenti attacchi da noi implementati: Label Poisoning Attack, Sponge Attack e Noise Injection.
Infine, proponiamo possibili direzioni future per il miglioramento delle difese contro tali minacce, basandoci sull'elenco delle difese allo stato dell'arte discusse anch'esse all'interno del documento.

Introduzione

Centralized Machine Learning e i suoi limiti

L'apprendimento automatico (Machine Learning - ML) è il processo di sviluppo e creazione di algoritmi e modelli capaci di apprendere attraverso l'addestramento su un dataset, per poi applicare la conoscenza acquisita su dati non ancora visti. Questo avviene grazie all'applicazione di varie operazioni sui dati, utilizzando diversi modelli come reti neurali, alberi decisionali, e molte altre tecniche di apprendimento automatico.

Nei sistemi classici di Machine Learning Centralizzato (Centralized Machine Learning - CML), i dati utilizzati per l'addestramento non vengono generati direttamente sul dispositivo che ospita il modello, ma provengono da molteplici fonti, come sensori IoT, smartphone e altri dispositivi distribuiti. Questi dati vengono raccolti e inviati a un server centrale, dove avviene il processo di addestramento del modello. Una volta che il modello è stato addestrato, può essere utilizzato per fare previsioni su nuovi dati che vengono successivamente inviati al server.
Tuttavia, questo approccio presenta diverse limitazioni che possono renderlo inadeguato in molti scenari moderni.

I limiti del Machine Learning Centralizzato includono problemi di privacy (poiché i dati devono essere inviati a un server centrale), scalabilità (elevati costi di trasferimento e archiviazione), sicurezza (rischio di attacchi informatici e single point of failure) e generalizzazione (modelli meno adatti a contesti distribuiti e diversificati).
Per superare questi ostacoli, è emerso il Federated Learning (FL), che consente di addestrare i modelli direttamente sui dispositivi locali, mantenendo i dati privati e inviando solo aggiornamenti dei parametri al server centrale.

Introduzione al Federated Learning (FL)

Il Federated Learning (FL) rappresenta un approccio innovativo all'apprendimento automatico collaborativo, che consente a più dispositivi o organizzazioni di partecipare al processo di addestramento di un modello globale senza condividere direttamente i propri dati sensibili. Questa caratteristica lo rende particolarmente adatto a settori in cui la privacy dei dati è fondamentale, come la sanità, la finanza e l'Internet of Things industriale (IIoT). Tuttavia, nonostante il suo potenziale, il FL presenta significative vulnerabilità derivanti dalla sua natura distribuita.

Ciclo di vita dei FL

Il ciclo di vita di un qualsiasi FL, può essere descritto in 6 passaggi fondamentali:

1. **Offerta e selezione dei partecipanti –** Il fornitore di servizi pubblica un compito di apprendimento e seleziona i partecipanti in base alle risorse offerte e ai requisiti richiesti.

2. **Trasmissione del modello globale –** Il server invia il modello globale iniziale ai partecipanti, specificando le condizioni per l'addestramento.

3. **Addestramento locale –** Ogni partecipante aggiorna il modello localmente con i propri dati, quindi invia le modifiche al server.

4. **Aggregazione e aggiornamento –** Il server aggrega gli aggiornamenti ricevuti per migliorare il modello globale e lo redistribuisce ai partecipanti per nuovi cicli di apprendimento.
   Per aggregare i pesi dei modelli locali aggiornati dai partecipanti e migliorare il modello globale, utilizza la tecnica Federated Averaging (FedAvg)
   La formula principale di FedAvg è la seguente:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k$$

Dove:

- w_{t+1} è il nuovo modello globale aggiornato,
- K è il numero totale di partecipanti (client),
- w_k è il modello locale aggiornato del client k,
- n_k è il numero di esempi di training posseduti dal client k,
- $n = \sum_{k=1}^{K} n_k$ è il numero totale di esempi di training aggregati.

5. **Distribuzione incentivi –** Quando il modello raggiunge le prestazioni desiderate, il server conclude l'addestramento e distribuisce incentivi ai partecipanti.

6. **Pubblicazione del modello –** Il modello finale viene pubblicato online per essere utilizzato dagli utenti in applicazioni pratiche.

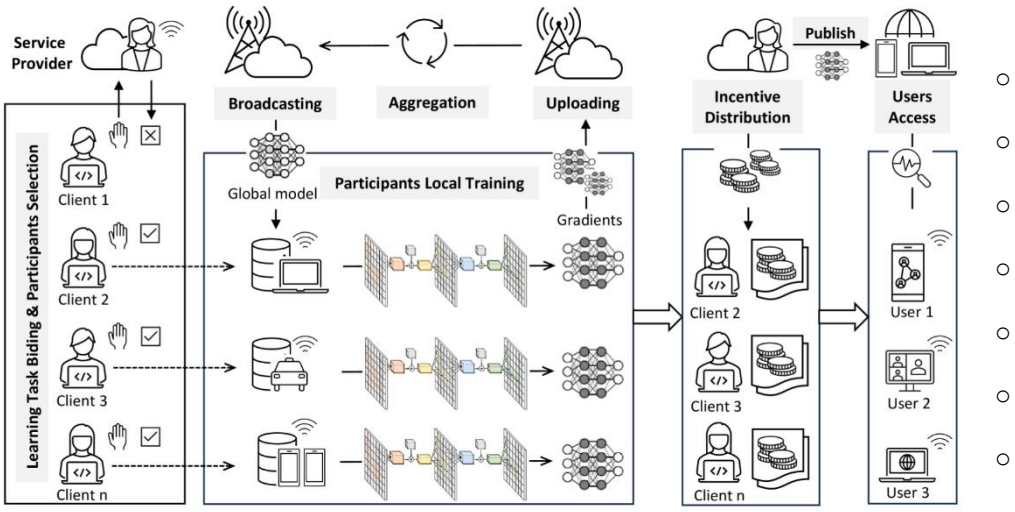

Immagine 1.1: Ciclo di vita dei Federated Learning

Tipologie di FL (HFL, VFL, FTL; CFL vs DFL)

Il Federated Learning (FL) può essere categorizzato in base alla distribuzione dei dati dei partecipanti o all'architettura del sistema, e riteniamo che queste classificazioni siano essenziali in quanto gli attacchi vengono effettuati anche a seconda che un FL sia di una topologia o di un'altra, dal momento che alcuni attacchi risultano più efficaci su un tipo rispetto che a un altro.

In base alla distribuzione dei dati:

Il FL può essere classificato come horizontal federated learning (HFL), vertical federated learning (VFL) e federated transfer learning (FTL).

- **HFL (Horizontal Federated Learning)** è applicabile a casi in cui i partecipanti possiedono campioni di dati differenti ma gli stessi spazi delle caratteristiche. Ad esempio, nella ricerca sulle malattie rare, gli ospedali possono avere pazienti distinti, ma le loro cartelle cliniche contengono caratteristiche simili.
- **VFL (Vertical Federated Learning)**: In questo caso, i partecipanti si sovrappongono negli spazi dei campioni, ma differiscono negli spazi delle caratteristiche. Ad esempio, banche e operatori di telecomunicazioni di una città possono tenere traccia degli stessi cittadini, ma i loro record contengono caratteristiche diverse.

- **FTL (Federated Transfer Learning)**: Integra i concetti di FL e transfer learning, in cui il transfer learning sfrutta la conoscenza acquisita da un dominio per migliorare le prestazioni in un altro dominio. L'FTL affronta la sfida dei campioni e delle caratteristiche limitate sovrapposte, addestrando un modello su un ampio dataset pubblico.

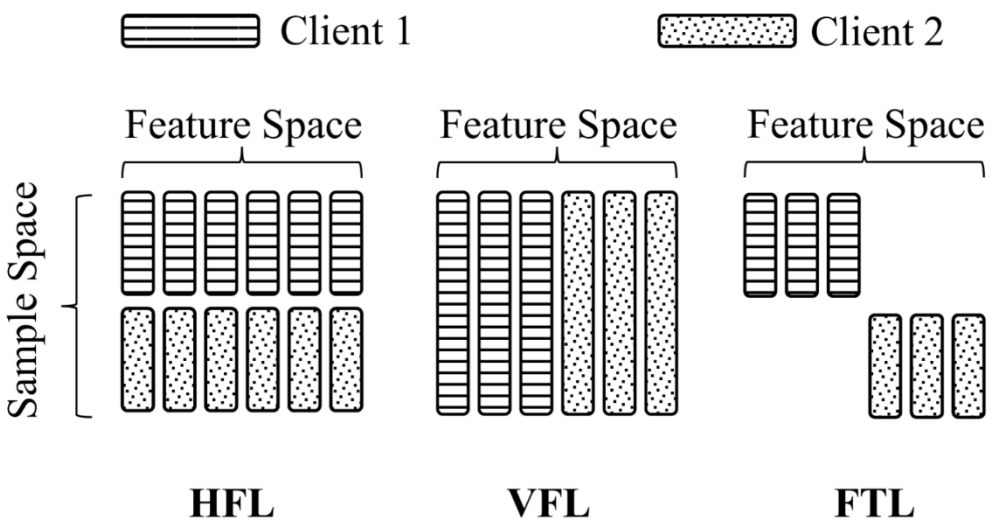

Immagine 1.2: Distribuzione dati dei client in HFL, VFL e FTL

In base all'architettura del sistema:

Il FL può essere suddiviso in Centralized Federated Learning (CFL) e Decentralized Federated Learning (DFL).

- **CFL (Centralized Federated Learning):** In questo caso, un server organizza l'intero processo di apprendimento, compresa la trasmissione dei modelli, la raccolta e l'aggregazione.
- **DFL (Decentralized Federated Learning):** A differenza del CFL, il DFL non coinvolge un server centrale. Seguendo una regola preimpostata, i partecipanti possono scambiarsi direttamente i loro modelli ed eseguire automaticamente l'aggregazione.

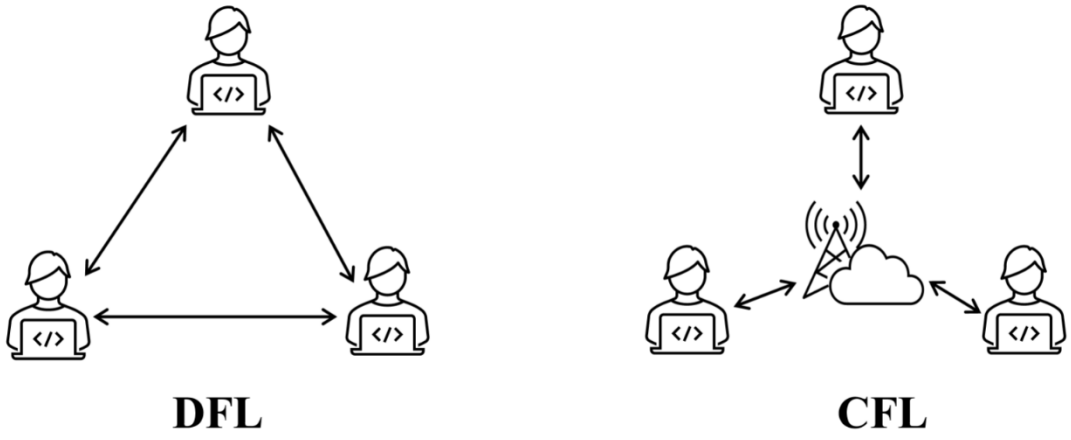

Immagine 1.3: Architetture dei sistemi CFL e DFL

Minacce e vulnerabilità dei FL

Iniziamo identificando ed enumerando le potenziali minacce e problematiche per un sistema di Federated Learning (FL). Queste minacce verranno analizzate più approfonditamente quando parleremo degli attacchi a un FL nel paragrafo 3.

Il Federated Learning è vulnerabile a diverse minacce alla sicurezza e alla privacy a causa della sua natura distribuita. Le minacce possono provenire da attori malevoli interni (Insiders) o esterni (Outsiders), con obiettivi che vanno dal furto di dati alla manipolazione dell'addestramento. Le vulnerabilità emergono a causa della trasmissione di parametri dei modelli, della mancanza di un controllo centralizzato e della difficoltà nel garantire l'integrità dei contributi dei client.

Le principali vulnerabilità

Le principali vulnerabilità includono:

- **Scarsa protezione della privacy**: Nei FL, i dispositivi client non condividono i dati grezzi, ma trasmettono i gradienti o altri parametri del modello a un server centrale o ad altri dispositivi per l'aggregazione. Tuttavia, anche questi gradienti possono contenere informazioni sensibili sui dati di addestramento.
- **Dipendenza da server centrali**: Molte implementazioni del Federated Learning si basano su un server centrale per coordinare l'addestramento. Questo server diventa un single point of failure (SPOF), rendendolo vulnerabile a diversi attacchi e problemi operativi.
- **Scalabilità limitata**: Il FL coinvolge potenzialmente milioni di dispositivi, ciascuno con diverse capacità computazionali e di rete. Ciò comporta problemi di scalabilità e di efficienza della comunicazione.

Problemi di Sicurezza e Privacy

Un aspetto fondamentale nell'analisi delle vulnerabilità e degli attacchi nel Federated Learning è distinguere se rappresentino una minaccia per la sicurezza del sistema o per la privacy. L'impatto di questi attacchi varia in base al contesto applicativo: in alcuni casi, le violazioni della privacy e la perdita di dati

possono risultare più dannose rispetto ai rischi per la sicurezza del sistema, o viceversa.

Ad esempio, nei sistemi di Federated Learning applicati alla rilevazione delle frodi bancarie, un attacco alla privacy potrebbe portare alla fuga di dati sensibili sui clienti, come transazioni finanziarie e informazioni personali. Questo potrebbe favorire il furto d'identità o frodi su larga scala. D'altra parte, un attacco alla sicurezza potrebbe manipolare i modelli di rilevamento delle frodi, rendendoli inefficaci e consentendo transazioni fraudolente senza essere rilevate.

Protocolli e Canali di Comunicazione

Nel contesto dei Federated Learning, i protocolli e i canali di comunicazione rappresentano un aspetto critico della sicurezza complessiva del sistema, poiché sono il mezzo attraverso il quale vengono scambiati dati, aggiornamenti di modello e informazioni di controllo tra i partecipanti.
Se questi canali non sono adeguatamente protetti, diventa possibile per un aggressore esterno intercettare, modificare o persino iniettare messaggi malevoli nel flusso di comunicazione. Ad esempio, in assenza di una crittografia robusta end-to-end, i dati trasmessi possono essere facilmente soggetti ad attacchi di eavesdropping, in cui un malintenzionato si limita ad ascoltare il traffico di rete per raccogliere informazioni sensibili.
Tale vulnerabilità diventa ancora più critica in presenza di protocolli di comunicazione che non prevedono meccanismi di autenticazione dei messaggi o controlli di integrità, poiché ciò apre la porta ad attacchi di tipo man-in-the-middle. In questi scenari, l'aggressore non si limita a monitorare la comunicazione, ma si inserisce attivamente nel canale, modificando i messaggi in transito, alterando i parametri o persino replicando informazioni con l’obiettivo di indurre comportamenti errati nel modello aggregato.
Altre problematiche possono insorgere con attacchi di replay, in cui un messaggio legittimo, catturato in un determinato momento, viene successivamente reinviato per disturbare il normale funzionamento del protocollo, potenzialmente inducendo errori nell'aggiornamento del modello globale.

Server Malevolo e Algoritmo di Aggregazione

I server rivestono un ruolo essenziale nel Federated Learning, in quanto responsabili della distribuzione dei parametri iniziali del modello, della raccolta degli aggiornamenti inviati dai client e della diffusione del modello globale agli utenti selezionati. Questa centralità li rende uno degli elementi più sensibili dell'intero ecosistema FL.

In una tipica architettura FL, i server sono spesso ospitati nel cloud e, di conseguenza, possono diventare bersagli di attacchi DDoS. Inoltre, se compromessi, potrebbero monitorare tutti gli aggiornamenti dei gradienti inviati dai client durante le fasi di addestramento, alterando il processo di aggregazione e mettendo a rischio l'integrità dell'apprendimento. Per prevenire questi scenari, è fondamentale un monitoraggio continuo dei server per individuare e mitigare eventuali vulnerabilità.

Tali vulnerabilità possono aprire la strada a diversi tipi di attacchi, che verranno approfonditi nelle sezioni successive. Le minacce spaziano dalle azioni di inferenza, volte a estrarre informazioni riservate dai client (attacchi di inferenza), fino ad attacchi diretti che compromettono il modello globale (avvelenamento del modello, o model poisoning).

Un server malevolo può adottare due approcci principali per ricostruire i dati privati dei client:

- **Attacco passivo:** si limita ad analizzare gli aggiornamenti inviati dai client senza alterarne il contenuto.
- **Attacco attivo:** tenta di isolare i modelli individuali condivisi dalle vittime, al fine di condurre attacchi più invasivi e mirati.

Un altro aspetto fondamentale per la sicurezza del sistema FL è **l'algoritmo di aggregazione** impiegato dal server per combinare gli aggiornamenti dei client e migliorare il modello globale. Questo componente rappresenta la prima barriera di protezione contro anomalie provocate da partecipanti malevoli che cercano di alterare il processo di apprendimento.

Tipo di attaccante

Come descritto nell'introduzione alle minacce e vulnerabilità, le minacce possono provenire da attori malevoli interni (Insiders) o esterni (Outsiders)

Gli attacchi insider sono generalmente considerati più pericolosi rispetto agli outsider, per i seguenti motivi:

- Gli insider hanno già accesso privilegiato al sistema, quindi possono danneggiarlo in modo più diretto e profondo, ad esempio manipolando direttamente i parametri del modello o influenzando l'intero processo di addestramento.
- Gli attacchi insider sono più difficili da rilevare, poiché l'attaccante fa parte del sistema e può mascherarsi da un partecipante legittimo.
- Gli insider possono compromettere la sicurezza dei dati e alterare il comportamento del modello, con potenziali conseguenze molto gravi a lungo termine.

Gli attacchi outsider sono comunque una minaccia significativa, ma solitamente sono limitati dal fatto che l'attaccante non ha accesso diretto al sistema. Tuttavia, se i canali di comunicazione non sono sicuri, gli outsider possono ancora causare danni come intercettare dati sensibili o modificare le comunicazioni, come in accordo con il paragrafo precedente sulle vulnerabilità dovute ai protocolli e canali di comunicazione.

Quando possono avvenire gli attacchi

Tali minacce possono verificarsi in qualsiasi fase del ciclo di vita del servizio di FL (paragrafo 1.3), dai processi di selezione dei partecipanti all'aggregazione dei modelli, poiché la sua architettura decentralizzata introduce nuove superfici di attacco rispetto al machine learning tradizionale.

- **Fase di Raccolta dei Dati (Data Collection Phase)**: I dati rimangono sui dispositivi degli utenti, senza essere centralizzati. Questo comporta la possibilità che utenti malevoli manipolino i loro dati senza essere rilevati.

- **Fase di Addestramento Locale (Local Training Phase):** Ogni dispositivo addestra un modello in modo indipendente senza che il server centrale possa verificare direttamente l'affidabilità dell'addestramento.

- **Fase di Aggregazione del Modello (Model Aggregation Phase)**: Il server centrale raccoglie gli aggiornamenti dai client e li aggrega, ma non ha una visione completa dei dati originali né un controllo diretto sulla qualità degli aggiornamenti ricevuti.

- **Fase di Distribuzione del Modello (Model Distribution Phase)**: Il modello aggiornato viene distribuito ai client tramite una rete non sempre sicura, esponendo il sistema ad attacchi di intercettazione.

- **Fase di Inferenza (Inference Phase)**: Il modello addestrato viene usato per fare previsioni, ma un attaccante può sfruttare questa fase per dedurre informazioni sui dati utilizzati per l'addestramento.

Analisi delle strategie di attacco nello stato dell'arte

In questa sezione, enumereremo e modelleremo una tassonomia degli attacchi sui sistemi FL, basata sulle vulnerabilità e minacce che abbiamo analizzato nel paragrafo 2

Gli attacchi nel contesto del FL possono compromettere l'integrità del modello, ridurne le prestazioni o violare la privacy dei partecipanti ed è per tale motivazione che la classificazione principale degli attacchi si basa su questa distinzione di obbiettivo.
Possiamo quindi distinguere 3 principali categorie di attacco:

1. **Attacchi alla Privacy** – Mirano a estrarre informazioni sensibili dai dati locali dei partecipanti.

2. **Attacchi alla Sicurezza del Modello** – Tentano di manipolare il modello in modo malevolo.

3. **Attacchi alla Robustezza e Disponibilità** – Puntano a ridurre l'efficienza o l'integrità dell'apprendimento federato.

Nella successiva parte del report presenteremo in maniera più dettagliata queste categorie di attacco introducendo gli attacchi più significativi di ognuna.

Attacchi alla Privacy

Introduzione

Gli attacchi alla privacy mirano a **estrarre informazioni sensibili** dai dati di addestramento dei partecipanti senza compromettere direttamente il funzionamento del modello. Il FL è vulnerabile a questi attacchi perché, sebbene i dati non vengano centralizzati, gli aggiornamenti del modello trasmessi tra client e server possono contenere informazioni utili per un attaccante.

Vulnerabilità sfruttate

- **Gradienti correlati ai dati**: I gradienti trasmessi possono rivelare informazioni sulle caratteristiche dei dati locali.
- **Modelli troppo confidenti**: Il modello può memorizzare informazioni specifiche sui dati, permettendo inferenze sulla loro appartenenza.

- **Mancanza di crittografia end-to-end**: Se i parametri sono trasmessi in chiaro, possono essere intercettati.

Principali attacchi alla privacy

1. **Gradient Inversion Attack** → Ricostruisce i dati originali dai gradienti.
2. **Membership Inference Attack (MIA)** → Determina se un dato è stato usato per addestrare il modello.
3. **Property Inference Attack** → Estrae caratteristiche generali dei dati di un nodo (es. genere, età media).

Gradient Inversion Attack

Il **Gradient Inversion Attack** è un attacco mirato alla privacy, in cui un attaccante utilizza i gradienti condivisi durante l'addestramento per ricostruire i dati originali dei partecipanti. Nel Federated Learning, i client calcolano i gradienti localmente sui propri dati e li inviano al server centrale per l'aggregazione, mantenendo apparentemente i dati privati. Tuttavia, l'attaccante può sfruttare la relazione matematica tra gradienti, pesi del modello e dati di input per risolvere un problema di ottimizzazione inversa, riuscendo così a ricostruire i dati che hanno generato quei gradienti. Questo attacco è più efficace in condizioni specifiche, come con modelli semplici, batch di dati piccoli o conoscenze pregresse sull'origine dei dati. Le conseguenze sono significative: i dati sensibili dei partecipanti possono essere compromessi, con impatti gravi soprattutto in settori che richiedono una forte protezione della privacy, come la sanità. Per mitigare questo rischio, si possono adottare tecniche come l'aggiunta di rumore ai gradienti (ad esempio attraverso la privacy differenziale) o l'uso di crittografia avanzata come quella omomorfica.

Membership Inference Attack

Il **Membership Inference Attack** è un attacco che mira a determinare se un determinato dato è stato utilizzato durante l'addestramento di un modello. Questo tipo di attacco sfrutta la capacità di un attaccante di interrogare il modello e analizzarne il comportamento, come la probabilità di output o i gradienti, per identificare se un dato specifico fa parte del dataset di

addestramento. L'attacco si basa sull'osservazione che i modelli tendono a rispondere in modo più sicuro o con maggiore accuratezza sui dati di addestramento rispetto a quelli mai visti, creando un divario tra i due casi che può essere sfruttato.

Le conseguenze di un Membership Inference Attack sono significative, in quanto la presenza di un dato specifico in un dataset potrebbe rivelare informazioni sensibili. Questo è particolarmente problematico in contesti come la sanità o la finanza, dove sapere che un dato paziente è stato utilizzato in un dataset potrebbe indicare la sua associazione a una particolare condizione medica o situazione economica.

Property Inference Attack

Il **Property Inference Attack** è un attacco che mira a estrarre informazioni globali o statistiche specifiche sul dataset utilizzato per addestrare un modello di machine learning, anche se tali proprietà non sono direttamente correlate al compito di apprendimento del modello. L'attaccante analizza il modello condiviso o i suoi parametri per dedurre caratteristiche sensibili del dataset, come la distribuzione demografica, la presenza di dati con determinate proprietà o altre informazioni aggregate. Questo è possibile perché i modelli spesso incorporano, in modo implicito, dettagli statistici sui dati di addestramento, anche se non necessari per il loro obiettivo primario.

Le conseguenze di un Property Inference Attack possono essere critiche in ambiti in cui il dataset è sensibile. Ad esempio, in un contesto medico, l'attaccante potrebbe inferire la proporzione di pazienti con una determinata condizione, violando la privacy collettiva. Difendersi da questo attacco richiede misure come l'introduzione di tecniche di Differential Privacy, che aggiungono rumore per mascherare le informazioni statistiche, e l'uso di modelli con maggiore robustezza, progettati per ridurre la perdita di informazioni non necessarie.

Attacchi alla Sicurezza del Modello

Gli attacchi all'integrità del modello mirano a manipolare i risultati del Federated Learning, inducendo il modello a imparare comportamenti errati o intenzionalmente alterati. Questo può avvenire modificando i dati di addestramento locali o manipolando gli aggiornamenti del modello.

Vulnerabilità sfruttate

- **Modello aggiornato dai client:** Poiché il modello viene aggiornato da nodi distribuiti, un nodo malevolo può inviare aggiornamenti manipolati.
- **Aggregazione non robusta:** Se il server FL usa una semplice media per aggregare i pesi dei client, un attaccante può influenzare significativamente il modello globale.
- **Mancanza di verifica sui contributi:** Il server non sempre è in grado di distinguere tra aggiornamenti legittimi e malevoli.

Principali attacchi all'integrità

1. **Backdoor Attack** → Inserisce comportamenti nascosti nel modello (es. riconoscere un trigger specifico).
2. **Poisoning Attack** → Modifica i dati locali per influenzare il modello globale.

Backdor Attack

Un **Backdoor Attack** è un attacco in cui un attaccante inserisce intenzionalmente una "porta sul retro" nel modello di apprendimento federato. L'idea principale dietro questo tipo di attacco è quella di addestrare il modello in modo tale che, pur mantenendo una buona performance generale su dati "normali", il modello risponda in modo errato o desiderato quando viene presentato con dati specifici, chiamati **trigger**. Questi trigger sono pattern o caratteristiche particolari che il modello non ha imparato in modo naturale, ma che sono stati intenzionalmente "insegnati" all'interno di un attacco backdoor.

Poisoning Attack

Il Poisoning Attack nel contesto del Federated Learning rappresenta una minaccia diretta all'integrità del modello globale, in quanto l'obiettivo dell'attaccante è compromettere la qualità e l'affidabilità del processo di apprendimento. Questo tipo di attacco può essere declinato in due modalità principali:

- **Data Poisoning**
- **Model Poisoning**

che, pur appartenendo entrambi alla categoria degli attacchi all'integrità, operano su livelli differenti del processo di apprendimento.

Nel caso del Data Poisoning, l'attacco avviene a monte, intervenendo direttamente sui dati di addestramento utilizzati da ciascun client. In un sistema di Federated Learning, ogni partecipante possiede un proprio dataset e utilizza questi dati per addestrare un modello locale. Se un attaccante riesce a modificare o a corrompere questi dati—ad esempio, alterando le etichette, inserendo esempi fuorvianti o manipolando la distribuzione delle feature—il modello locale imparerà informazioni distorte rispetto alla realtà. Quando questi modelli, basati su dati alterati, vengono inviati al server centrale per l'aggregazione, il risultato è un modello globale che rispecchia e amplifica tali anomalie. Questo può tradursi in previsioni errate o in bias sistematici, con conseguenze potenzialmente gravi soprattutto in ambiti critici come la sanità o la sicurezza.

Il Model Poisoning, invece, interviene in una fase successiva, agendo direttamente sugli aggiornamenti del modello che vengono comunicati dal client al server. In questo scenario, l'attaccante potrebbe, ad esempio, controllare uno o più dispositivi partecipanti e inviare aggiornamenti alterati, manipolando i pesi o i gradienti del modello. Anche se i dati locali di addestramento rimangono integri, gli aggiornamenti manipolati possono distorcere il processo di aggregazione a livello globale. Tecniche di Model Poisoning possono includere l'amplificazione dei gradienti, in modo tale che il contributo del nodo malevolo diventi sproporzionato rispetto a quello degli altri client, o l'inserimento di backdoor, ossia parametri che innescano comportamenti specifici e indesiderati quando vengono presentati input particolari. La difficoltà principale nel contrastare il Model Poisoning risiede nel fatto che, a differenza del Data Poisoning, il problema non riguarda la qualità intrinseca dei dati, ma la legittimità e la coerenza degli aggiornamenti inviati.

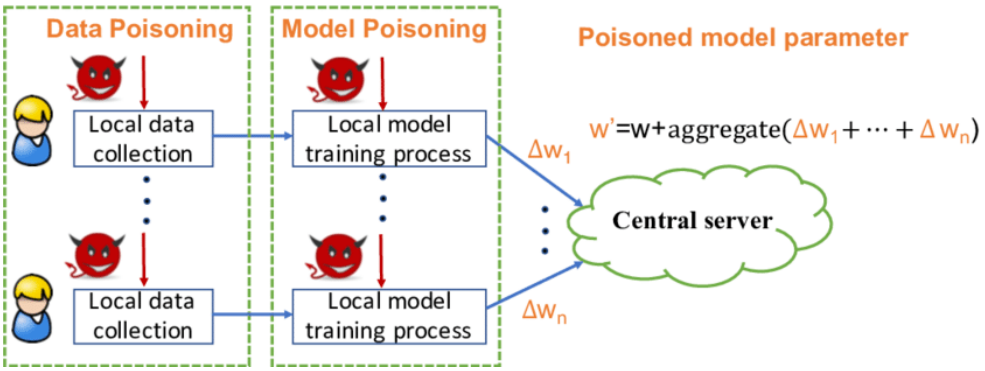

Immagine 4.1: Data Poisoning vs Model Poisoning

Attacchi alla Robustezza e Disponibilità

Introduzione

Gli attacchi alla disponibilità mirano a compromettere il normale funzionamento del Federated Learning, rallentando o interrompendo l'addestramento. A differenza degli attacchi all'integrità, che mirano a modificare il modello, questi attacchi si concentrano sul disturbare il processo di apprendimento stesso.

Vulnerabilità sfruttate

- **Partecipazione aperta:** In FL, molti client possono partecipare, e un attaccante può infiltrarsi con nodi malevoli.
- **Mancanza di un sistema di validazione dei client:** Il server potrebbe non essere in grado di distinguere i contributi utili da quelli dannosi.
- **Dipendenza da più nodi:** Se un attacco coinvolge un numero sufficiente di nodi, può bloccare l'addestramento.

Principali attacchi alla disponibilità

1. **Free-riding Attack** → Un nodo partecipa al training senza contribuire realmente, ma usufruisce del modello finale.
2. **Sybil Attack** → Un attaccante crea più identità false per ottenere un peso maggiore nel processo di aggregazione.
3. **Sponge Attack** → Un nodo invia parametri completamente casuali per degradare le prestazioni del modello.

Free-riding Attack

L'attacco di Free-riding è un comportamento opportunistico in cui un nodo partecipa al processo di training di un modello federato, ma senza contribuire realmente con dati utili o aggiornamenti

validi. Il nodo, in sostanza, si limita a ricevere il modello aggiornato senza offrire nulla in cambio.

Questo tipo di attacco è particolarmente pericoloso perché consente a entità malintenzionate di ottenere i benefici dell'apprendimento federato senza dover investire risorse computazionali o dati. Un problema ancora più grave è che, se molti nodi adottano questa strategia, l'addestramento complessivo del modello potrebbe risultare meno efficace, dato che le informazioni condivise con il server centrale saranno limitate o addirittura inutili.

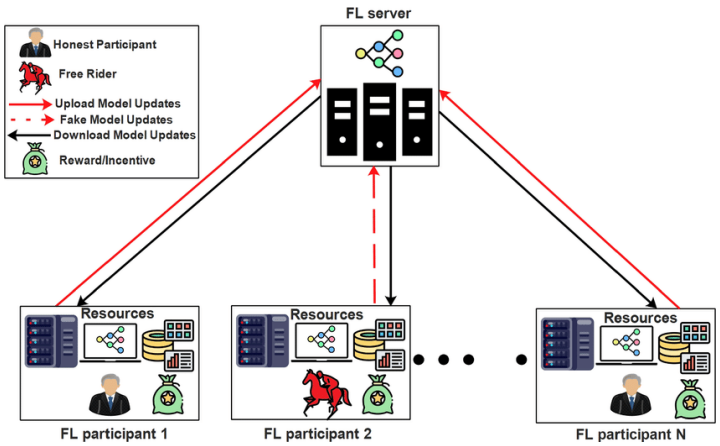

Immagine 4.2: Free-riding Attack

Sybil Attack

Il Sybil Attack è una tecnica in cui un attaccante crea e registra più identità false (nodi Sybil) per ottenere un'influenza sproporzionata nel processo di aggregazione del modello. Questo tipo di attacco sfrutta il fatto che, nel Federated Learning, il modello viene aggiornato in base ai contributi dei partecipanti: se un singolo attore riesce a controllare molteplici nodi, può manipolare il risultato finale del training.

Gli effetti di un Sybil Attack possono essere:

- **Manipolazione del modello**: Un attaccante potrebbe alterare il comportamento del modello per renderlo meno accurato o per introdurre bias specifici.
- **Compromissione della privacy**: L'attaccante potrebbe usare i nodi falsi per raccogliere informazioni sensibili da altri partecipanti.

- **Interruzione dell'apprendimento:** Se molti nodi Sybil inviassero dati errati o avvelenati, il modello potrebbe non convergere mai a una soluzione ottimale.

Sponge Attack

Lo Sponge Attack è una tecnica in cui un nodo invia parametri completamente casuali al modello con l'obiettivo di degradarne le prestazioni e aumentare il consumo di risorse computazionali. Il termine "sponge" (spugna) fa riferimento al fatto che l'attaccante assorbe inutilmente la capacità di elaborazione del sistema, rallentandone il funzionamento.

L'attacco può avere diversi effetti negativi:

- **Aumento dei costi computazionali:** Il server dovrà elaborare aggiornamenti inutili, con un maggiore impiego di tempo ed energia.
- **Diminuzione della qualità del modello:** Se i parametri casuali venissero accettati, il modello potrebbe imparare informazioni errate e peggiorare le sue prestazioni.
- **Rallentamento dell'addestramento:** Se il server deve gestire un numero elevato di aggiornamenti spazzatura, il tempo di convergenza del modello può aumentare drasticamente.

Attacchi ai FL da un’altra prospettiva

Oltre alla classificazione basata sull'obiettivo dell'attacco (privacy, sicurezza del modello, robustezza/disponibilità), possiamo categorizzare gli attacchi nel Federated Learning (FL) secondo altre prospettive che verranno introdotte nei paragrafi successivi

Tipi di attacco in base alle tipologie del FL:

La tipologia di Federated Learning (FL) e l'architettura del sistema (Centralized vs Decentralized, descritte nei paragrafi precedenti, influenzano direttamente il tipo di attacchi che è possibile eseguire e le difese necessarie. Ogni tipo di FL ha delle peculiarità che possono renderlo più vulnerabile a certi tipi di attacchi rispetto ad altri, sempre tenendo conto dell’obbiettivo dell’attaccante.

1. Attacchi in base alla distribuzione dei dati:

a) Horizontal Federated Learning (HFL):

Poiché i partecipanti in HFL condividono le stesse caratteristiche (ad esempio, gli ospedali hanno dati clinici simili) ma dati differenti (pazienti diversi), un attaccante potrebbe sfruttare le seguenti vulnerabilità:

- Attacchi a Gradienti: In HFL, un partecipante malintenzionato potrebbe inviare aggiornamenti di gradienti falsificati (ad esempio, "avvelenamento dei gradienti"), che potrebbero influenzare il modello globale. Se il partecipante possiede una conoscenza sufficiente dei dati simili agli altri partecipanti (es. malattia rara comune a più ospedali), potrebbe manipolare i gradienti per fare in modo che il modello globale diventi meno preciso o più indirizzato verso una determinata malattia.
- Model Poisoning (avvelenamento del modello): L'attaccante può inviare un modello manipolato, che ha comportamenti errati (ad esempio, etichettando erroneamente pazienti) e contaminare l'intero processo di aggregazione.

b) Vertical Federated Learning (VFL):

Poiché in VFL i partecipanti possiedono gli stessi campioni (es. i dati dei cittadini), ma caratteristiche diverse (es. banca vs telecomunicazioni), un attacco mirato potrebbe essere:

- Data Leakage (fuoriuscita di dati): Poiché i partecipanti possiedono i dati degli stessi individui, un attaccante potrebbe sfruttare questa sovrapposizione per tentare di "svelare" dati sensibili da altre fonti, creando attacchi incrociati tra i partecipanti.
- Inference Attacks: In un contesto VFL, dove gli stessi campioni sono condivisi ma le caratteristiche variano, un attaccante potrebbe cercare di ricavare informazioni private sugli utenti (es. profili finanziari da una banca) attraverso le inferenze sui dati combinati.

c) Federated Transfer Learning (FTL):

In FTL, l'attaccante potrebbe approfittare della sovrapposizione limitata tra i dati e sfruttare il dataset pubblico di supporto:

- Poisoning Attack via Transfer Learning: Se un attaccante ha accesso ai dati di un dominio simile (ad esempio, un dominio pubblico), potrebbe manipolare il modello di base prima che venga trasferito ai partecipanti, compromettendo l'accuratezza o la privacy.
- Malicious Knowledge Transfer: L'attaccante potrebbe cercare di trasferire conoscenze errate o false da un dominio esterno, causando danni nel dominio target (ad esempio, la compagnia di assicurazioni che addestra il modello su dati contaminati).

2. Attacchi in base all'architettura del sistema:

Gli attacchi variano anche in base a come è strutturato il sistema FL, ossia se si utilizza un server centrale (CFL) o se il sistema è decentralizzato (DFL).

a) Centralized Federated Learning (CFL):

Nel CFL, dove un server centrale coordina tutte le operazioni (transmissione del modello, raccolta dei dati e aggregazione), ci sono diverse vulnerabilità:

- Server Poisoning Attack: Un attaccante potrebbe cercare di compromettere il server centrale, manipolando l'intero processo di aggregazione o infettando i modelli che vengono distribuiti ai partecipanti.
- Model Inversion Attack: Con l'accesso al server centrale, l'attaccante potrebbe tentare di recuperare i dati sensibili o risalire a informazioni private (ad esempio, tentando di

ricostruire i dati di addestramento attraverso il modello globale).

- Man-in-the-Middle Attacks: Durante la trasmissione dei modelli tra server e partecipanti, un attaccante potrebbe intercettare e manipolare i modelli globali, causando malfunzionamenti nel processo di aggregazione.

b) Decentralized Federated Learning (DFL):

Nel DFL, dove i partecipanti collaborano direttamente senza un server centrale, l'architettura decentralizzata porta a diversi tipi di vulnerabilità:

- Sybil Attack: Poiché nel DFL non c'è un server centrale per validare i partecipanti, un attaccante potrebbe creare più identità false (Sybil attack) e partecipare al processo di aggregazione, con l'intento di compromettere la qualità del modello finale.
- Byzantine Attack: In un sistema decentralizzato, un partecipante malintenzionato potrebbe inviare informazioni errate o dannose durante l'aggregazione del modello, influenzando negativamente il risultato finale. Questo è un tipo di attacco noto come "attacco bizantino", dove le comunicazioni tra partecipanti vengono manipolate deliberatamente per causare disaccordo nel modello globale.
- Eavesdropping and Reverse Engineering: Poiché nel DFL i partecipanti si scambiano modelli direttamente tra loro, un attaccante potrebbe intercettare i modelli durante la comunicazione e tentare di inserire vulnerabilità nel processo, oppure cercare di estrarre informazioni sensibili dai modelli scambiati.

Nella mente di un attaccante

L'obiettivo principale di questo documento, oltre a fornire una base teorica sui Federated Learning, le loro vulnerabilità, i possibili attacchi e le relative difese, è dimostrare concretamente il funzionamento di tali attacchi e i loro effetti sui FL. Tuttavia, prima di affrontare questo aspetto, riteniamo sia utile adottare la prospettiva di un potenziale attaccante per comprendere le motivazioni alla base di questi attacchi e valutare quali siano più o meno efficaci in relazione all'obiettivo che si intende raggiungere.

Perché attaccare un FL

Se assumessimo la prospettiva di un attaccante, le motivazioni che potrebbero spingerci a colpire un sistema di Federated Learning sarebbero molteplici, variando in base agli obiettivi che si desidera raggiungere.
Abbiamo già analizzato e classificato alcun
Tra i più comuni, possiamo individuare:

1. Guadagno Finanziario

- **Furto di dati sensibili:** Sebbene FL preservi la privacy dei dati locali, i modelli possono contenere informazioni sensibili. Un attaccante potrebbe cercare di estrarre tali informazioni tramite attacchi di inferenza, per poi rivenderle o usarle in altre attività illecite.
- **Manipolazione dei modelli per trarre profitto:** Un attaccante potrebbe cercare di manipolare il modello addestrato per favorire decisioni specifiche che lo avvantaggiano economicamente, ad esempio truccando un modello predittivo per favorire un investimento o una scommessa.

2. Sabotaggio

- **Corruzione del modello globale (Data Poisoning):** Un attaccante potrebbe voler danneggiare la performance del sistema compromettendo il modello globale. In questo caso, l'attacco di poisoning (iniezione di dati errati) sarebbe finalizzato a degradare la qualità del modello, facendolo meno efficace.
- **Danno alla reputazione di una piattaforma:** Se un FL viene utilizzato da un’organizzazione per applicazioni critiche,

sabotare i modelli potrebbe danneggiarne la fiducia da parte degli utenti e la sua reputazione.

- **Attacco ideologico o etico:** In alcuni casi, gli attacchi possono essere motivati da questioni etiche o politiche, come la critica al modo in cui i dati vengono gestiti o utilizzati, spingendo l'attaccante a compromettere il sistema come forma di protesta.

3. Vulnerabilità dei Sistemi di Apprendimento

- **Esplorare le vulnerabilità del sistema:** Gli attaccanti potrebbero voler testare le difese di un sistema FL per scoprire vulnerabilità, come nel caso di attacchi di backdoor, dove il modello viene manipolato in modo subdolo.
- **Attacchi mirati per l'inversione del gradiente (Model Inversion):** Un attaccante potrebbe cercare di ottenere accesso ai dati sensibili utilizzati nel processo di addestramento sfruttando vulnerabilità nei gradienti condivisi tra i partecipanti al sistema.

4. Privacy e Spionaggio

- **Accesso a informazioni private senza violare direttamente i dati:** Gli attaccanti potrebbero essere motivati dal desiderio di raccogliere dati sensibili senza dover violare le leggi sulla privacy. Con tecniche di attacco avanzate come il *Model Inversion* o *Membership Inference*, l'attaccante può cercare di dedurre informazioni sui dati di addestramento senza avere accesso diretto a essi.
- **Spionaggio aziendale:** Se il FL fosse utilizzato per addestrare modelli in ambito industriale, un attaccante potrebbe cercare di raccogliere informazioni sulla ricerca o sui processi aziendali sensibili.

5. Concorrenza Sleale

- **Guadagno competitivo:** In un contesto commerciale, un attaccante potrebbe cercare di compromettere i modelli di Federated Learning di una compagnia rivale per ottenere un vantaggio competitivo, abbassando la qualità dei modelli predittivi della concorrenza o accedendo a tecniche di apprendimento avanzate.

7. Accesso a Modelli Avanzati

- **Rubare modelli avanzati:** Gli attaccanti potrebbero essere motivati dal desiderio di accedere a modelli avanzati

addestrati con Federated Learning. Tali modelli, soprattutto in settori come la medicina, la finanza, o l'automotive, possono contenere tecniche altamente sofisticate o informazioni.

In accordo con gli attacchi descritti nel capitolo precedente, ad ognuna di queste motivazioni corrisponderà un attacco conforme con esse, ad esempio per un furto di dati corrisponderà un attacco alla privacy, o per un sabotaggio un attacco all'integrità del FL.

Difendere la Sicurezza e la Privacy dei FL:

Nel contesto del Federated Learning (FL), gli attacchi possono compromettere la privacy, l'integrità e la disponibilità del modello globale come in accordo con i paragrafi precedenti.
Le metodologie difensive vengono implementate per proteggere il sistema dalle minacce note e dalle vulnerabilità, riducendo la probabilità che gli attacchi possano sfruttarle.
Possiamo distinguere due tipi di difese: proattive e reattive.

- Una **difesa proattiva** implica l'adozione di metodi economicamente sostenibili in un ambiente di produzione per individuare anticipatamente le minacce e i rischi correlati.
- Una **difesa reattiva**, invece, consiste nell'identificare un attacco in corso e adottare misure di risposta come parte del processo di mitigazione.

Nelle prossime sezioni verranno presentate le difese più comuni sia proattive che reattive.

Difese Proattive

Le tecniche proattive si basano su diversi principi:

- **Protezione della privacy** - Impedire che gli attaccanti estraggano informazioni sensibili dai dati locali.
- **Robustezza dell'aggregazione** - Evitare che aggiornamenti malevoli influenzino il modello globale.
- **Sicurezza hardware** - Isolare e proteggere l'elaborazione dei dati nei dispositivi locali.
- **Controllo degli accessi e policy di sicurezza** - Regolare chi può partecipare all'addestramento federato.

Tecniche basate sulla crittografia e privacy

Queste tecniche mirano a proteggere sia i dati originali che i parametri del modello durante il processo di addestramento federato. L’obiettivo è impedire a potenziali attaccanti di estrarre informazioni sensibili dai dati condivisi, pur consentendo una collaborazione efficace tra le entità partecipanti.

Differential Privacy (DP)

Principi di funzionamento:

La Differential Privacy introduce un livello di rumore controllato nei dati o nei gradienti (i cambiamenti calcolati durante l'addestramento) prima che questi vengano trasmessi al server. L'aggiunta di questo rumore è calibrata tramite due parametri principali:

- **Epsilon (ε):** Determina il grado di privacy: valori più bassi garantiscono una maggiore protezione ma possono ridurre la precisione del modello.
- **Delta (δ):** Indica la probabilità residua che la garanzia di privacy non sia completamente rispettata.

In sostanza, DP garantisce che la presenza o l'assenza di un singolo dato nel dataset non influenzi significativamente il risultato finale, rendendo difficile l'identificazione di individui specifici.

Implicazioni:

- **Privacy vs. Accuratezza:** L'aggiunta di rumore può degradare leggermente la qualità dei dati, riducendo la precisione del modello, ma offre una protezione robusta contro attacchi inferenziali come il Membership Inference Attack.
- **Composizione e budget di privacy:** In scenari con più iterazioni di addestramento, è necessario gestire un "budget di privacy" per mantenere un livello di protezione accettabile.

Secure Multi-Party Computation (SMPC)

Principi di funzionamento:
SMPC permette a più partecipanti di calcolare una funzione comune sui loro dati senza doverli rivelare reciprocamente. Ciò viene realizzato mediante la suddivisione dei dati in "condivisioni" o frammenti, che vengono distribuiti e utilizzati per eseguire operazioni crittografate. Al termine del calcolo, i risultati parziali vengono combinati per ottenere l'output desiderato, mantenendo i dati originali nascosti.

Implicazioni:

- **Sicurezza:** Garantisce che ogni partecipante non abbia accesso ai dati degli altri, riducendo il rischio di violazioni della privacy.
- **Overhead computazionale:** L'uso di protocolli SMPC, per quanto estremamente sicuri, introduce un significativo sovraccarico computazionale e di comunicazione, rendendoli meno adatti per applicazioni con risorse limitate o in tempo reale.

Crittografia Omomorfica

Principi di funzionamento:
La crittografia omomorfica consente di eseguire operazioni matematiche direttamente su dati cifrati. Il risultato, una volta decrittato, corrisponde esattamente al risultato che si sarebbe ottenuto eseguendo le stesse operazioni sui dati in chiaro. Questo consente di mantenere i dati protetti per l'intera durata del calcolo.

Implicazioni:

- **Massima sicurezza:** Il server o il provider cloud non visualizza mai i dati in chiaro, proteggendo contro furti e attacchi inferenziali.
- **Costi computazionali:** L'elaborazione di dati cifrati in modalità omomorfica è estremamente intensiva in termini di risorse e tempo, rendendola adatta principalmente a scenari dove la sicurezza ha la priorità assoluta rispetto alla velocità.

Secure Aggregation

Principi di funzionamento:
Nel federated learning, il protocollo di Secure Aggregation permette al server centrale di combinare gli aggiornamenti dei modelli provenienti dai vari client senza accedere ai singoli contributi. I dati inviati dai client sono crittografati in modo che solo l'aggregato possa essere decrittato.

Implicazioni:

- **Riduzione del rischio di inversione dei gradienti:** Poiché il server non vede gli aggiornamenti individuali, è più difficile per un attaccante risalire ai dati originali.

- **Scalabilità:** La tecnica è particolarmente efficace in scenari con un alto numero di partecipanti, dove l'aggregazione di molte fonti aumenta la difficoltà di identificare dati specifici.

Tecniche basate sulla robustezza dell'aggregazione

Queste metodologie sono progettate per proteggere il processo di addestramento da aggiornamenti malevoli o anomali, garantendo che il modello globale non venga compromesso da attacchi mirati a manipolare i dati.

Robust Aggregation (Krum, Median, Trimmed Mean)

Principi di funzionamento:

- **Krum:** Questo algoritmo seleziona uno o pochi aggiornamenti "affidabili" basandosi sulla distanza (ad esempio, euclidea) tra gli aggiornamenti forniti dai client. Gli aggiornamenti che risultano troppo divergenti vengono scartati.
- **Median e Trimmed Mean:** Questi metodi calcolano rispettivamente la mediana o una media escludendo una certa percentuale di valori estremi (outlier). L'idea è quella di minimizzare l'effetto di eventuali contributi malintenzionati o errati.

Implicazioni:

- **Protezione contro Model Poisoning e Backdoor Attacks:** Questi algoritmi permettono di mitigare il danno causato da aggiornamenti intenzionalmente manipolati, preservando la qualità del modello finale.
- **Trade-off tra robustezza e reattività:** Mentre escludere aggiornamenti anomali aumenta la sicurezza, può anche portare a una riduzione della velocità di adattamento del modello ai nuovi dati.
-

Byzantine Fault Tolerance (BFT)

Principi di funzionamento:
BFT si ispira ai protocolli di consenso utilizzati nei sistemi distribuiti (ad esempio, nelle blockchain) e garantisce il corretto funzionamento del sistema anche in presenza di nodi (client) che agiscono in modo arbitrario o malevolo. Questi protocolli prevedono meccanismi di verifica e consenso, in cui la maggioranza dei partecipanti deve concordare sul risultato finale, ignorando i contributi anomali.

Implicazioni:

- **Resilienza in ambienti ostili:** BFT consente al sistema di mantenere performance e coerenza anche se una parte dei client è compromessa.
- **Complessità e latenza:** I protocolli BFT possono introdurre una certa latenza e complessità computazionale, specialmente in sistemi con molti partecipanti.

Tecniche basate sulla sicurezza hardware

Queste soluzioni sfruttano componenti fisici e tecnologie a livello di hardware per creare ambienti sicuri in cui eseguire operazioni critiche, proteggendo i dati anche in caso di compromissione del software.

Trusted Execution Environments (TEE)

Principi di funzionamento:
I TEE creano una "zona sicura" all'interno del processore (o di un chip dedicato) in cui vengono eseguiti calcoli e operazioni crittografiche. Tecnologie come Intel SGX e ARM TrustZone offrono ambienti isolati, dove il codice e i dati sono protetti anche se il sistema operativo principale viene compromesso.

Implicazioni:

- **Isolamento delle operazioni sensibili:** I dati elaborati all'interno di un TEE non sono accessibili all'esterno, aumentando significativamente la sicurezza contro attacchi software.
- **Limitazioni hardware:** L'adozione dei TEE richiede hardware compatibile e può comportare delle restrizioni in termini di risorse computazionali disponibili per l'esecuzione dei calcoli sicuri.

Tecniche basate su policy e gestione degli accessi

Queste tecniche si concentrano sull'istituzione di regole, protocolli e sistemi di autenticazione che regolano chi e quali dispositivi possono partecipare al processo di federated learning, riducendo il rischio di accessi non autorizzati.

Controllo degli accessi

Principi di funzionamento:
Il controllo degli accessi prevede l'uso di meccanismi di autenticazione (come certificati digitali, token crittografici o

liste di dispositivi approvati) per garantire che solo entità verificate possano inviare aggiornamenti o accedere ai dati del modello. Queste misure riducono la possibilità che attori malevoli possano partecipare al processo.

Implicazioni:

- **Sicurezza rafforzata:** Limitando la partecipazione solo a dispositivi fidati, si riduce il rischio di attacchi provenienti da fonti esterne o compromesse.
- **Gestione amministrativa:** È necessario implementare sistemi di gestione delle identità e aggiornare regolarmente le liste dei dispositivi autorizzati, con un impatto sulla complessità gestionale del sistema.

Politiche di sicurezza per i dispositivi FL

Principi di funzionamento:
Le politiche di sicurezza definiscono un insieme di regole e standard per la gestione dei dispositivi partecipanti al federated learning. Queste politiche possono includere:

- **Verifiche periodiche del software:** Assicurarsi che i dispositivi utilizzino versioni aggiornate e sicure dei software necessari.
- **Controllo della presenza di malware:** Implementazione di sistemi di rilevamento e prevenzione contro software dannosi.
- **Requisiti hardware e configurazioni di sicurezza:** Standard che specificano le caratteristiche minime di sicurezza che ogni dispositivo deve rispettare per partecipare.

Implicazioni:

- **Riduzione dei rischi:** L'adozione di politiche rigorose limita l'accesso a dispositivi vulnerabili, riducendo il rischio di compromissione del modello.
- **Coordinamento centralizzato:** La gestione delle policy richiede un sistema centralizzato di monitoraggio e aggiornamento, che garantisca la conformità di tutti i partecipanti.

Difese Reattive

Le tecniche reattive si focalizzano sulla rilevazione e risposta a minacce o attacchi già in corso, cercando di mitigare i danni e ripristinare la sicurezza del sistema in tempo reale.

- **Rilevamento degli attacchi** → Identificare attività sospette che possano indicare la presenza di attacchi in corso.
- **Recupero rapido** → Minimizzare il danno a lungo termine, ripristinando velocemente il sistema a uno stato sicuro.
- **Adattamento dinamico** → Modificare il comportamento del sistema in tempo reale per contrastare le minacce emergenti.
- **Monitoraggio continuo** → Tenere sotto controllo costante il sistema per rilevare anomalie in tempo reale.

Tra le principali difese reattive:

Anomaly Detection

Principi di funzionamento:

Il rilevamento delle anomalie si basa su modelli statistici o di machine learning per identificare comportamenti che si discostano da quelli normali, suggerendo la presenza di attività malevoli. Gli algoritmi di rilevamento esaminano variabili come i pattern di aggiornamento del modello, i cambiamenti nei gradienti e le comunicazioni tra i nodi.

Implicazioni:

- **Rilevamento tempestivo:** Il sistema può individuare rapidamente attacchi come quelli di "model poisoning" o "data poisoning", bloccando la minaccia prima che causi danni significativi.
- **Falsi positivi:** I modelli possono generare falsi positivi, identificando erroneamente attività legittime come sospette, richiedendo una regolazione fine per ridurre tali errori.
- **Dinamismo:** Gli algoritmi di rilevamento devono adattarsi continuamente ai cambiamenti nei modelli legittimi di comportamento.

Rollback e Ripristino

Principi di funzionamento:

Nel caso di attacchi riusciti, una tecnica di rollback consente di annullare gli aggiornamenti malevoli, ripristinando il sistema a uno stato precedente noto come sicuro. Questo processo di recupero può avvenire su vari livelli, dal livello dei dati a quello dei modelli.

Implicazioni:

- Sicurezza immediata: Il rollback consente di ridurre rapidamente i danni, annullando gli effetti di un attacco senza compromettere l'intero sistema.
- Perdita di dati: Il ripristino può comportare la perdita di dati recenti, il che potrebbe ridurre l'efficacia del modello nel rispondere ai dati nuovi.
- Semplicità vs complessità: Sebbene il rollback sia efficace, la gestione delle versioni e il monitoraggio dei cambiamenti possono aumentare la complessità operativa.

Gli attacchi da noi implementati

In questo capitolo analizzeremo gli attacchi selezionati per essere applicati al sistema di Federated Learning. Nello specifico, gli attacchi considerati sono tre: **Noise Injection**, **Label Poisoning Attack** e **Sponge Attack**. Per ciascun attacco è stato esaminato il comportamento del modello globale in presenza di uno, tre e sei client malevoli.

Infine, in accordo con quanto descritto nel capitolo precedente, descriveremo brevemente quali difese possono contenere o annullare tali attacchi.

Label Poisoning Attack

Il Label Poisoning Attack è un tipo di attacco di avvelenamento dei dati (data poisoning) che mira a compromettere il processo di apprendimento di un modello modificando intenzionalmente le etichette dei dati di addestramento. Questo attacco è particolarmente efficace in Federated Learning (FL), dove il controllo centralizzato sui dati è limitato e ogni nodo addestra il modello sui propri dati locali.

Abbiamo quindi scambiato i label per ogni campione all'interno del dataset, così da generare, nella fase di test, delle predizioni errate affinché venga riconosciuta una classe sbagliata.

```
if self.mode == "label" and round_num > ROUND:
    # Trova le righe che corrispondono a [1., 0., 0.]
    mask1 = (labels == torch.tensor([1., 0., 0.], dtype=DTYPE)).all(dim=1)
    # Trova le righe che corrispondono a [0., 1., 0.]
    mask2 = (labels == torch.tensor([0., 1., 0.], dtype=DTYPE)).all(dim=1)
    # Trova le righe che corrispondono a [0., 0., 1.]
    mask3 = (labels == torch.tensor([0., 0., 1.], dtype=DTYPE)).all(dim=1)
    if i==0:
        # Sostituisci [1., 0., 0.] con [0., 1., 0.] e viceversa e [0., 0., 1.] con [1., 0., 0.]
        labels[mask1] = torch.tensor([0., 1., 0.], dtype=DTYPE)
        labels[mask2] = torch.tensor([1., 0., 0.], dtype=DTYPE)
        labels[mask3] = torch.tensor([1., 0., 0.], dtype=DTYPE)
    elif i==1:
        # Sostituisci [1., 0., 0.] con [0., 0., 1.] e viceversa e [0., 0., 1.] con [1., 0., 0.]
        labels[mask1] = torch.tensor([0., 0., 1.], dtype=DTYPE)
        labels[mask2] = torch.tensor([0., 0., 1.], dtype=DTYPE)
        labels[mask3] = torch.tensor([1., 0., 0.], dtype=DTYPE)
    else:
        # Sostituisci [0., 1., 0.] con [0., 0., 1.] e viceversa e [0., 0., 1.] con [0., 1., 0.]
        labels[mask1] = torch.tensor([0., 1., 0.], dtype=DTYPE)
        labels[mask2] = torch.tensor([0., 0., 1.], dtype=DTYPE)
        labels[mask3] = torch.tensor([0., 1., 0.], dtype=DTYPE)
```

Immagine 6.1: Codice Label Poisoning Attack

Le etichette che andremo a modificare identificano tre classi: silence, aluminium e steel. Ogni elemento dei tensori segue la struttura [x, y, z], dove un valore pari a 1 in una delle tre posizioni indica l'appartenenza alla classe corrispondente, mentre gli altri due valori sono impostati a 0. Ad esempio, se x = 1, il dato appartiene alla classe silence, mentre y = 1 indica aluminium e z = 1 corrisponde a steel.

In questo attacco, manipoliamo le etichette alterando la disposizione di zeri e uni in modo sistematico. Nello specifico, per ogni elemento del dataset, le classi vengono scambiate secondo uno schema prestabilito. Ad esempio, durante la prima iterazione di training:

- Gli elementi inizialmente assegnati alla classe silence vengono rietichettati come aluminium e viceversa.
- Gli elementi appartenenti alla classe steel vengono invece riassegnati alla classe silence.

Questo processo continua iterativamente, seguendo uno schema di rotazione per alterare le etichette a ogni fase di addestramento.

Di seguito, viene riportato un grafico che mostra l'evoluzione dell'accuratezza del modello a seguito di questa manipolazione

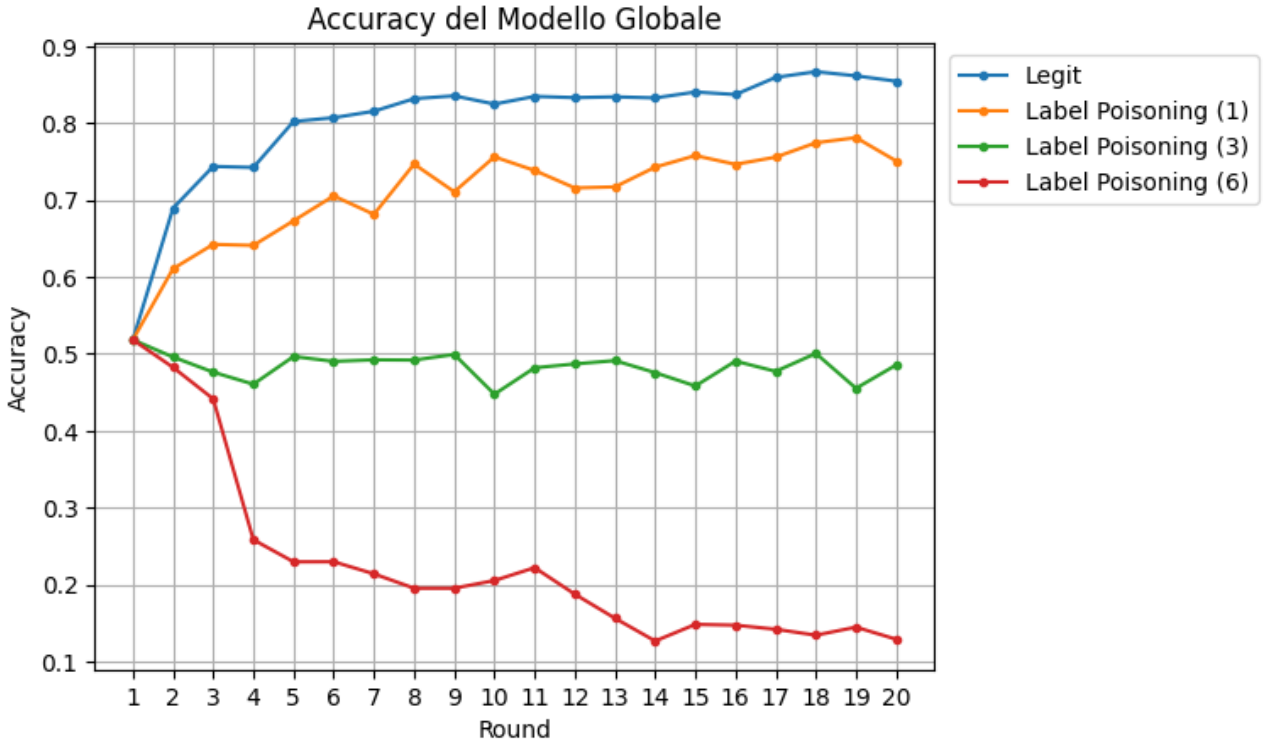

Immagine 6.2: Accuracy Label Poisoning

La curva di colore **blu** rappresenta il comportamento del sistema in assenza di attacchi, fornendo un riferimento per le prestazioni del modello non compromesso. Le altre tre curve, disposte in ordine decrescente, mostrano invece l'accuratezza del modello quando è sottoposto a un **attacco di Label Poisoning**, rispettivamente con **1, 3 e 6 client malevoli**.

Come previsto, all'aumentare del numero di client compromessi, l'accuratezza del modello diminuisce progressivamente, evidenziando l'impatto negativo dell'attacco sulla capacità del sistema di effettuare classificazioni corrette.

Sponge Attack

La teoria dello Sponge Attack è stata già trattata nei capitoli precedenti; pertanto, qui ci concentreremo esclusivamente sull'implementazione del nostro attacco. Per il secondo attacco, sostituiremo gli input forniti al modello durante la fase di training con tensori di grandi dimensioni generati casualmente. Inoltre, per ogni nuovo input, verrà assegnata un'etichetta anch'essa casuale. In questo modo, non solo le prestazioni sia del modello globale che dei modelli locali tenderanno a diminuire, ma, a causa delle dimensioni dei tensori di input, la fase di training richiederà più tempo del previsto.

```python
if self.mode == "sponge" and round_num > ROUND:
    n_poison = 8192
    # Crea tensori grandi con valori casuali (consumo memoria)
    sponge_data = torch.randn(
        (n_poison, *TENSOR_SIZE),
        dtype=DTYPE,
        device=DEVICE)
    # Sostituisci gli input originali
    inputs = sponge_data

    # Assegna etichette casuali ai dati avvelenati
    labels = torch.randint(
        0, 3,
        (n_poison, labels.shape[1]),
        device=DEVICE
    ).to(DTYPE)
```

Immagine 6.3: Codice Sponge Attack

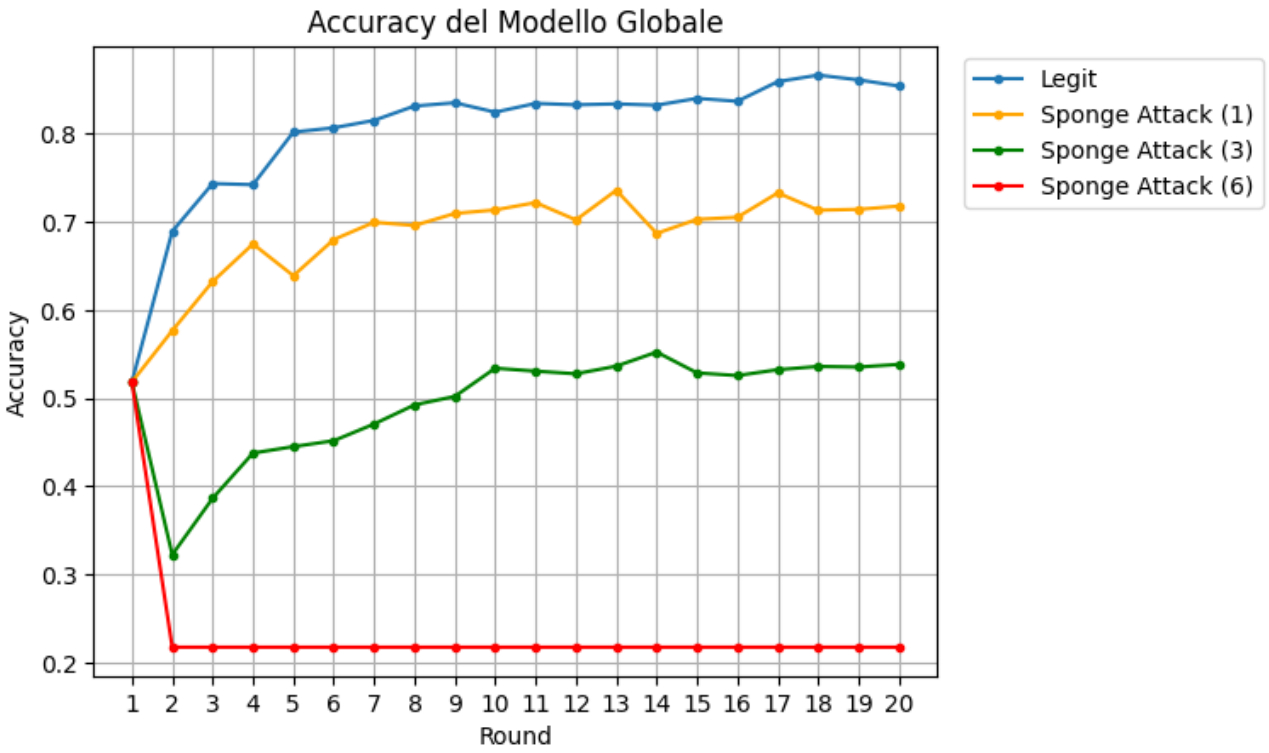

Immagine 6.4: Accuracy Sponge Attack

I colori che descrivono le quattro funzioni rimangono gli stessi del grafico precedente. Il comportamento è simile a quello del label poisoning con 1 e 3 client malevoli, mentre quando tutti e 6 i client sono malevoli si osserva un brusco calo delle prestazioni per lo Sponge Attack. Tuttavia, in questo caso, l'accuratezza tende ad allinearsi a un valore stabile, mentre nel Label Poisoning, con l'aumentare dei round, l'accuratezza continua a diminuire, raggiungendo il ventesimo round con un valore inferiore rispetto a quello dello Sponge Attack. Come già spiegato, lo Sponge Attack che abbiamo implementato non ha solo lo scopo di alterare i dati, ma anche di consumare molte risorse durante la fase di training, rallentandola significativamente. Di seguito, viene mostrato un grafico che evidenzia il tempo in cui il modello è sotto attacco rispetto al periodo in cui opera in modalità "legit".

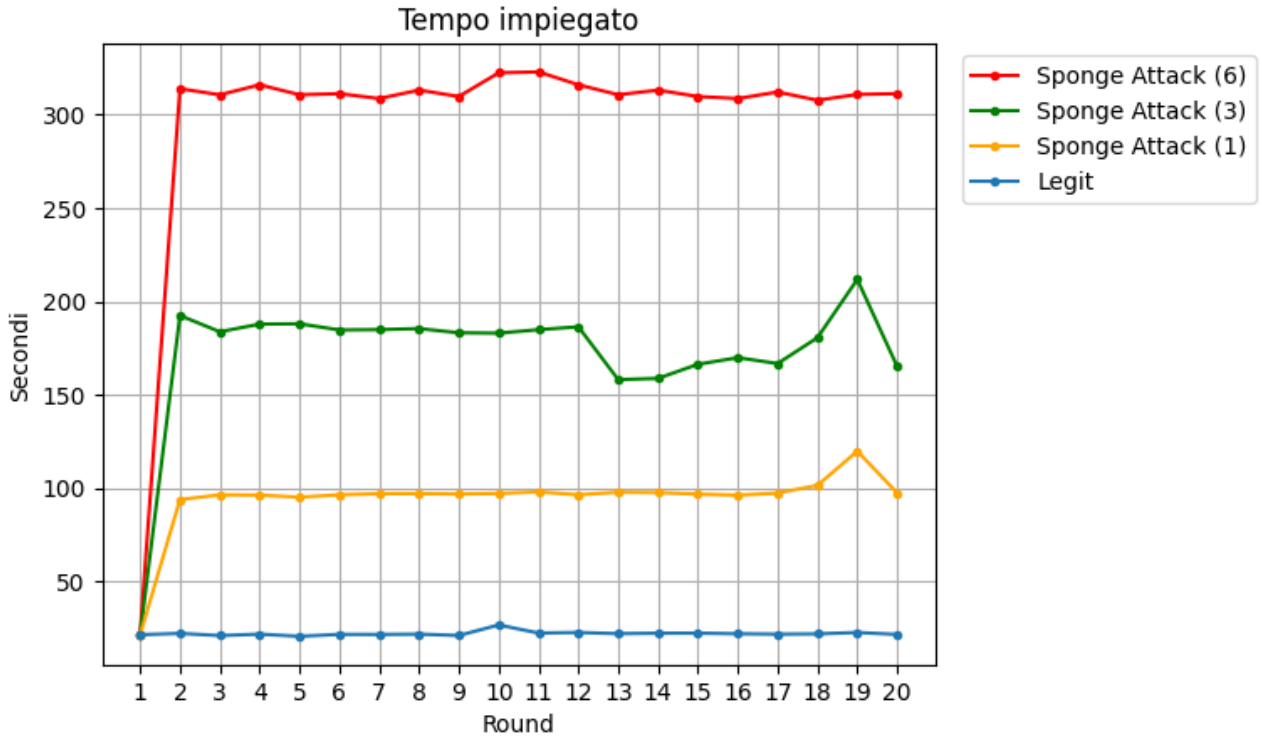

Immagine 6.5: Tempo di esecuzione Sponge Attack

Noise Injection

Il **Noise Injection Attack** è una tecnica malevola utilizzata per compromettere i modelli di **Federated Learning** (FL) inserendo del **rumore** (noise) nei dati di addestramento, nei gradienti o nei pesi del modello, con l'obiettivo di danneggiare il processo di apprendimento del modello globale. Questo attacco si distingue per il suo carattere subdolo, in quanto può essere difficile da rilevare e da contrastare, specialmente in un ambiente distribuito come il Federated Learning.

In un **Noise Injection** il rumore può essere aggiunto in modo casuale o seguire uno schema progettato per influenzare negativamente l'accuratezza del modello globale. L'intento dell'attacco è quello di degradare la performance del modello senza compromettere in modo evidente il processo di addestramento, rendendo difficile il rilevamento.

```python
if self.mode == "noise" and round_num > ROUND:
    for _, para in self.net.named_parameters():
        noise = torch.randn_like(para) * 0.5
        para.data += noise # Aggiunta di rumore ai pesi
        para.data *= (torch.rand(1).item() * 2)  # Amplificazione casuale
        noise = torch.randn_like(para.grad.data) * 0.5
        para.grad.data += noise # Aggiunta di rumore ai gradienti
        para.grad.data *= (torch.rand(1).item() * 2)  # Amplificazione casuale
```

Immagine 6.6: Codice del Weight and Gradient Noise Injection

Il codice riportato sopra genera un rumore sotto forma di un tensore con valori casuali di bassa entità e lo somma sia ai pesi che ai gradienti del modello, subito dopo che la funzione di perdita è stata definita. Di seguito, è mostrato un grafico che illustra l'evoluzione dell'accuratezza del modello.

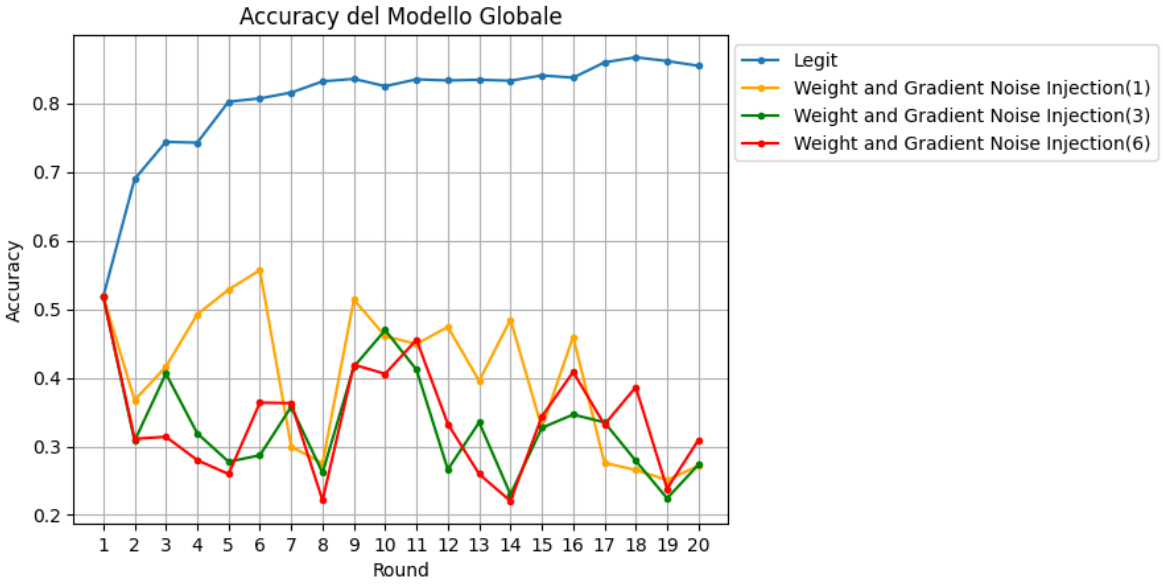

Immagine 6.7: Accuracy Noise Injection

La funzione di colore blu rappresenta il sistema quando non è sottoposto ad alcun attacco, mentre le altre tre funzioni, disposte dall'alto verso il basso, mostrano il comportamento del sistema in presenza di attacchi da parte di rispettivamente 1, 3 e 6 client malevoli. La particolarità di questo attacco risiede nel fatto che il modello riduce la sua accuratezza, raggiungendo valori simili indipendentemente dal numero di client coinvolti.

Questo risultato per quanto raro non è necessariamente errato per 3 motivazioni principali:

- **Saturazione:** Potrebbe esserci una sorta di "saturazione" nell'effetto del rumore sull'accuratezza. Oltre una certa

soglia di rumore (che potrebbe essere già raggiunta con un singolo client), l'aumento del rumore (aggiungendo client aggiuntivi) non peggiora ulteriormente l'accuratezza in modo significativo.

- **Compensazione:** È possibile che l'aggiunta di rumore da più client porti a una sorta di "compensazione" degli effetti negativi del rumore. In altre parole, il rumore aggiunto da client diversi potrebbe in qualche modo "annullarsi" a vicenda, portando a un andamento dell'accuratezza simile a quello con un singolo client.
- **Randomizzazione:** Aggiungendo rumore in modo casuale a gradienti e pesi, potrebbe succedere che un singolo client apporti modifiche drastiche a pesi o gradienti, mentre più client, pur essendo in maggioranza, introducono rumore in modo meno significativo.

Confronto tra i 3 attacchi

Tutti e tre i nostri attacchi influenzano negativamente l'accuratezza del modello globale, sebbene ciascuno presenti caratteristiche differenti. Si osserva che i secondi due attacchi seguono un andamento più lineare rispetto al primo. Inoltre, possiamo affermare che il *Weight and Gradient Noise Injection* risulta più efficace rispetto agli altri due quando è presente un singolo client malevolo, mentre, all'aumentare dei client malevoli, il *Label Poisoning* e lo *Sponge Attack* tendono a ridurre drasticamente l'accuratezza, raggiungendo un minimo globale di 0.12, valore attribuito dal *Label Poisoning*. Lo *Sponge Attack*, pur collocandosi a metà strada tra gli altri due in termini di accuratezza, ha l'obiettivo anche di compromettere la disponibilità del sistema.

Conclusioni:

Il Federated Learning (FL) rappresenta un approccio innovativo e promettente per l'apprendimento automatico distribuito, offrendo vantaggi significativi in termini di privacy e scalabilità rispetto ai tradizionali metodi di Machine Learning centralizzato. Tuttavia, come abbiamo dimostrato in questo lavoro, il FL non è immune a minacce e vulnerabilità, che possono compromettere la sicurezza, la privacy e l'efficacia del modello globale.

Attraverso l'analisi delle principali strategie di attacco, abbiamo evidenziato come gli attaccanti possano sfruttare le vulnerabilità intrinseche del FL per compromettere l'integrità del modello, violare la privacy dei dati o ridurre la disponibilità del sistema. In particolare, abbiamo implementato e valutato tre attacchi specifici: Label Poisoning Attack, Sponge Attack e Noise Injection, dimostrando come ciascuno di essi possa influenzare negativamente le prestazioni del modello, con impatti diversi in base al numero di client malevoli coinvolti.

I risultati ottenuti mostrano che:

- Il **Label Poisoning Attack** è particolarmente efficace nel ridurre l'accuratezza del modello, soprattutto quando il numero di client malevoli aumenta.
- Lo **Sponge Attack**, oltre a degradare le prestazioni del modello, ha l'obiettivo di rallentare il processo di addestramento, consumando risorse computazionali e compromettendo la disponibilità del sistema.
- Il **Noise Injection** ha un impatto più subdolo, ma comunque significativo, sull'accuratezza del modello, con effetti che tendono a stabilizzarsi indipendentemente dal numero di client malevoli.

Per contrastare queste minacce, abbiamo discusso diverse strategie difensive, sia proattive che reattive, tra cui l'uso di tecniche di crittografia avanzata (come la Differential Privacy e la Secure Multi-Party Computation), algoritmi di aggregazione robusta (come Krum e Trimmed Mean), e meccanismi di rilevamento delle anomalie. Tuttavia, è importante sottolineare che nessuna difesa è perfetta, e un approccio multilivello che combini diverse tecniche è essenziale per garantire la sicurezza e la privacy dei sistemi FL.

In conclusione, il Federated Learning offre un potenziale enorme per applicazioni in settori critici come la sanità, la finanza e l'Internet of Things, ma richiede un'attenzione particolare alla sicurezza e alla privacy. La ricerca futura dovrebbe concentrarsi sullo sviluppo di difese più avanzate e sull'integrazione di meccanismi di sicurezza fin dalle prime fasi di progettazione dei sistemi FL, al fine di mitigare le minacce emergenti e garantire un'applicazione sicura ed efficace di questa tecnologia.