Network-Level Adversaries in Federated Learning

Giorgio Severi*, Matthew Jagielski†, Gökberk Yar*, Yuxuan Wang‡, Alina Oprea*, Cristina Nita-Rotaru*

* Northeastern University, Khoury College of Computer Sciences
† Google Research
‡ LinkedIn Corporation

Abstract—Federated learning is a popular strategy for training models on distributed, sensitive data, while preserving data privacy. Prior work identified a range of security threats on federated learning protocols that poison the data or the model. However, federated learning is a networked system where the communication between clients and server plays a critical role for the learning task performance. We highlight how communication introduces another vulnerability surface in federated learning and study the impact of network-level adversaries on training federated learning models. We show that attackers dropping the network traffic from carefully selected clients can significantly decrease model accuracy on a target population. Moreover, we show that a coordinated poisoning campaign from a few clients can amplify the dropping attacks. Finally, we develop a server-side defense which mitigates the impact of our attacks by identifying and up-sampling clients likely to positively contribute towards target accuracy. We comprehensively evaluate our attacks and defenses on three datasets, assuming encrypted communication channels and attackers with partial visibility of the network.

I. INTRODUCTION

Federated Learning (FL) or collaborative learning, introduced by McMahan et al. [1], has become a popular method for training machine learning (ML) models in a distributed fashion. In FL, a set of clients perform local training on their private datasets and contribute their model parameters to a centralized server. The server aggregates the local model parameters into the global model, which is then disseminated back to the clients, and the process continues iteratively until convergence. Training ML models using FL allows the server to take advantage of a large amount of training data, and each client to retain the privacy of their data. FL can be deployed in either a cross-silo setting with a small number of participants available in all rounds of the protocol, or in a cross-device setting with a large number of participants, which are sampled by the server and participate infrequently in the protocol.

Recent work studying the security of FL has highlighted the risk of attacks by compromised clients through data poisoning [2] and model poisoning [3], [4] under different objectives such as availability, targeted, and backdoor attacks. While availability objectives aim at compromising the accuracy of the model indiscriminately [5], [6], targeted and backdoor attacks only affect a specific subset of samples [4], [7], [8] and are harder to detect. Among the proposed defenses, availability attacks can be mitigated by gradient filtering [5], [6], [9], while targeted and backdoor attacks could be addressed by clipping gradient norms at the server, during model updates, to limit the individual contributions to the global model [4], [7].

FL is a networked system where communication between server and clients plays a critical role as the global model is learned in rounds with contributions from multiple clients. Thus, communication represents another point of vulnerability that an attacker can exploit to influence the global model learned by the system. In this case, an attacker leverages network-level information and attack capabilities to prevent the ML algorithm from receiving the information needed to learn an accurate model. An attacker can exploit the specific communication channels for FL to perturb messages sent between server and clients via jamming [10], BGP hijacking [11], compromising routers [12], hosts [13], or transport-level protocols [14]. Network-level attacks have been shown to impact other systems such as Bitcoin [15], payment-channel networks [16], and connected cars [17]. These adversaries are especially relevant when political or economic incentives result in a government or company exploiting an AS for which sub-populations are geographically localized. For instance, this attack may be used to censor a word prediction model (especially targeting a country’s minority language), or modify words associated with a competitor company or unfavorable political movements.

In this paper, we conduct the first study of network-level adversaries’ impact on machine learning models trained with federated learning. We analyze the ways an attacker can exploit information sent over the network to maximally damage the learning algorithm under realistic conditions such as encrypted channels and partial visibility of the network. Specifically, we show that attackers who can carefully orchestrate dropping of the network traffic of selected clients can significantly decrease model accuracy on a target population. Our key insight is that a few clients contribute significantly to the accuracy of the model for target subpopulations, and we create an algorithm that allows the attacker to identify such clients, and, by interfering with only those clients, significantly outperform randomized packet dropping. Moreover, we show that model poisoning attacks from a small set of clients can further amplify the impact of the targeted dropping attacks. As an example, on a text classification problem, selectively dropping updates from only 5 clients from the target population can lead to a decrease in accuracy from 87% to 17%, on a particular target class. Furthermore, by adding a poisoning attack of equal size, the model accuracy decreases to 0% on the same target class. Finally, we consider a resource limited adversary, who can...
observe only a small subset of the participating clients, and show that our attacks are still effective, for instance inflicting a 43% relative accuracy drop when observing only a third of the clients in a computer vision task.

Federated learning system owners do not usually control the underlying network and might not be able to implement network-level mitigations to make communication more resilient. Complementary to such network-level defenses, we propose a server-level defense that is agnostic to how the dropping attack is performed. Our defense modifies client sampling in each round of FL training to increase the likelihood of selecting clients who sent useful updates in previous rounds of the protocol, while decreasing the probability of selecting less relevant clients. Interestingly, defensive client selection leverages the same procedure for client identification employed by the network-level adversary. The defense is extremely effective against a targeted dropping attack. For instance, in the same text classification task mentioned above, while an unmitigated attack would completely disrupt the model accuracy on the target task (from 87% to 17%), the defense achieves an accuracy of 96%, which exceeds that of the original model before an attack is mounted. Moreover, our defense can be combined with a standard poisoning defense based on update clipping to withstand both targeted dropping and model poisoning attacks. On the same task, the combined dropping and poisoning attack brings the target accuracy to 0, and the combination of our defense with clipping results in 94% accuracy on the target population. For encrypted communication the improvements compared to the original accuracy can be as high as 34%.

To summarize, the contributions of the paper are: (i) the first study of network-level adversaries in FL, specifically packet dropping attacks targeting a subpopulation and amplification by using model poisoning attacks; (ii) an algorithm for identification of highly contributing clients, allowing the attacker to be more effective than just randomly dropping traffic between clients and server; (iii) a defense based on up-sampling highly contributing clients to the learning task which can be combined with clipping to mitigate both targeted dropping and poisoning attacks; (iv) a comprehensive evaluation across multiple model architectures, datasets, and data modalities (image and text). For reproducibility, all code is publicly released.

II. BACKGROUND AND THREAT MODEL

A. Federated Learning

FL considers a set of $n$ clients, each with a local dataset $D_i$, and a server $S$. Clients train locally on their own datasets and the server requests model updates, rather than data, building an aggregated global model iteratively over time [18]. We consider here the Federated Averaging training algorithm designed for cross-device settings [1]. In each round $1 \leq t \leq T$, the server randomly selects a subset of $m \leq n$ clients to participate in training, and sends them the current global model $f_{t-1}$. Each selected client $i$ trains locally using dataset $D_i$, for a fixed number of $T_L$ epochs. The server updates the global model $f_t$ by mean aggregation of local updates $U_i$: $f_t = f_{t-1} + \frac{\sum_{i=1}^{m} U_i}{m}$. Data privacy can be further enhanced in FL by secure aggregation performed via Multi-Party Computation (MPC) [19].

B. Threat Model

Adversarial goal: An attacker can target the accuracy for all the classes of the learning task – availability attacks, or target a particular class – targeted attacks. Targeted attacks are much more difficult to detect as the attacker strives to make the model retain its test accuracy for non-targeted points to avoid trivial detection. We consider targeted attacks and define a population to be one of the classes in the learning task, but this notion could be extended to sub-classes as well. We do not consider poisoning availability attacks which are detectable and can be addressed with existing defenses [5], [6], [9].

Attack strategies: The attacker can conduct poisoning attacks, or network-level attack. Poisoning attacks can target the training data – an adversary injects maliciously crafted data points to poison the local model [2], or the model - where the adversary compromises a set of clients and sends malicious updates to the protocol with the goal of achieving a certain objective in changing model’s prediction [4], [7], [8]. Both these attacks are conducted by manipulating directly the inputs to the machine learning algorithm. We consider an attacker with model poisoning (POISON_MODEL) capability, using it to amplify network-level attacks.

In network-level attacks, based on their network-level capabilities, the adversary can observe communication sent over the network and prevent the machine learning algorithm from receiving the information needed to learn a good model. The basic action for a network-level attack is dropping traffic. We consider a smart attacker that selectively drops traffic to maximize the strength of the attack while minimizing detection. The attacker has several decisions to make: (1) what clients to select to drop their contributions, (2) when to start the attack given that federated learning is an iterative protocol, and (3) how many packets (i.e., local models) to drop. We focus on a targeted dropping attack in which the attacker selects a set of clients contributing highly to the target class for dropping their contributions. We compare that with an attack which drops the traffic of a subset of randomly chosen clients.

Adversarial network-level knowledge: We consider encrypted communication and partial network visibility scenarios for the communication in the FL protocol. For comparison we also consider unencrypted communication.

1) COMM.PLAIN. All communication between clients and server is unencrypted. This is a baseline scenario where a network-level adversary obtains maximum information. We do not expect this to happen in practice and we study this model to understand the maximum possible damage.

2) COMM.ENC. All communication between clients and server is encrypted. This is a realistic scenario, where the network-level adversary could infer the set of clients participating in each round, but not the exact model

\[1\text{https://github.com/ClonedOne/Network-Level-Adversaries-in-Federated-Learning}\]
updates they send. This is the typical deployment for FL, using IPSec, HTTPS, or application-level encryption.

3) **COMM_ENC LIMITED.** A special case of **COMM_ENC** where communication is encrypted, and the adversary is limited to only observe a fixed subset of the clients participating in the protocol. This is the most constrained attacker we consider.

**Adversarial global model knowledge:** Since cross-device FL is an open system, the adversary can always participate in the FL protocol as one of the clients. We assume that the adversary obtains the global model updates \( f_t \) at each round \( t \).

To summarize, we consider an adversary that has either the **COMM.PLAIN**, **COMM.ENC** or **COMM.ENC.LIMITED** network capability, has knowledge of the global model at each round, and targets a particular victim population of interest. We also consider an adversary that additionally has **POISON.MODEL** poisoning capabilities.

## C. Feasibility of Network-level Attacks for Federated Learning

FL protocols such as Federated Averaging [1] usually abstract the communication protocol between the server and clients. In practice, clients will communicate with the server either directly through a TCP connection, or through a multi-hop overlay network of hop-by-hop TCP or customized transport services. Finally, the last hop connecting the client to the Internet is often in the form of wireless communication. Network infrastructure and routing protocols facilitate all this communication through physical or logical connectivity. By exploiting the underlying routing protocols and network topology an attacker can position themselves to observe and interfere with data of interest; they can then further impact the accuracy of the global model and create an effect similar to that created through a data or model poisoning attack.

Packet dropping attacks can be achieved in multiple ways, and have been studied for network-level adversaries who perform physical-layer attacks in wireless networks [10], router compromise [12], or transport-level attacks [14]. Each of these methods will require different attacker capabilities. For example for physical layers attacks for wireless clients, the attacker needs to be in their proximity - this is both a powerful attack since jamming is usually difficult to defend against, and limiting since it will be difficult for the attacker to attack geographically distributed clients without significant resources. In the case of routing, while a router might be more difficult to compromise, in practice the impact can be bigger as it might have control over the traffic of multiple clients. Last, but not least, if customized overlays are used for communication, compromising nodes in the overlay is slightly easier as they are typically hosts, and the number of clients that can be impacted depends on the scalability of the service.

### III. NETWORK-LEVEL ATTACKS ON FEDERATED LEARNING

In this section we describe in detail our targeted network attacks against a population in federated learning.

### A. Identification of Highest-Contributing Clients

The key insight of our attack is that only a small number of clients contribute to the accuracy of a target population of interest. Thus, the most effective and difficult to detect strategy is to drop the traffic just for those highly contributing clients. However, the attacker will first need to identify such clients. We design a Client Identification algorithm aimed at determining the set of clients whose model updates lead to the largest improvements in target population accuracy. Our method, described in Algorithm 1, supports both plain and encrypted network communication, and is very effective even if the attacker has limited observability of the network. We demonstrate in Section V that if the adversary has high success at identifying the clients contributing to the task of interest, then the targeted dropping attack impact is much higher compared to random dropping.

**Algorithm 1: Loss Difference Client Identification**

**Data:** Target Population Dataset \( D^* \), loss function \( \ell \), rounds \( T_N \), count of clients to drop \( k_N \), visibility parameter \( v \).

**Function** clientIdentification \((D^*, \ell, T_N, k_N, v)\):

\[
f_0 = \text{GETGLOBALMODEL}(0) \quad \Delta = []
\]

for \( t \in [1, T_N] \) do

\[
M_t = \text{GETPARTICIPANTS}(v)
\]

\[
f_t = \text{GETGLOBALMODEL}(t)
\]

if **COMM.ENC** then

\[
\ell_{t} = \ell(f_{t-1}, D^*) - \ell(f_t, D^*)
\]

for \( j \in M_t \) do

\[
\Delta[j] = \text{CONCATENATE}(\Delta[j], [L_t])
\]

if **COMM.PLAIN** then

\[
f_t^j = \text{GETLOCALMODEL}(j)
\]

\[
\ell_{t}^j = \ell(f_{t-1}^j, D^*) - \ell(f_t^j, D^*)
\]

\[
\Delta[j] = \text{CONCATENATE}(\Delta[j], [L_{t}])
\]

// Compute the average loss change for each client

for \( j \in \Delta \) do

\[
\Delta[j] = \frac{1}{|M|} \sum \Delta[j]
\]

\[
Z = \text{GETLARGESTVALUES}(\Delta, k_N)
\]

return \( Z \)

The main intuition is that the adversary computes the loss of the model before and after the updates on the target class for a number of rounds \( T_N \). Rounds in which the loss decreases correspond to an increase in accuracy on the target class. The adversary tracks all the clients participating in those rounds, and computes a loss difference metric per client. In the plain communication case, the adversary will determine the set of clients whose model updates lead to the largest improvements in target population accuracy. Our method, described in Algorithm 1, supports both plain and encrypted network communication, and is very effective even if the attacker has limited observability of the network. We demonstrate in Section V that if the adversary has high success at identifying the clients contributing to the task of interest, then the targeted dropping attack impact is much higher compared to random dropping.

In more detail, to measure each client’s contribution to the target population, the adversary computes the loss difference between successive model updates on the target population (using a representative dataset \( D^* \) from the population):

\[
L_t^j = \ell(f_{t-1}^j, D^*) - \ell(f_t^j, D^*)
\]
Note that the second term \( \ell(f'_i, D^*) \) is the loss of the local update of client \( j \) in round \( t \) for \textsc{Comm Plain}, but becomes the loss of the global model \( \ell(f_i, D^*) \) when the adversary only has access to aggregated updates (\textsc{Comm ENC}, \textsc{Comm ENC Limited}). The value \( L'_i \) measures the decrease in the target class loss before and after a given round \( t \). To more reliably estimate client contributions, especially with aggregation of the model updates, the values of \( L'_i \) are accumulated over time. The adversary maintains a list for each client \( j \) of the values \( L'_i \) for rounds \( t \) in which it participated. This allows the adversary to update the list of clients most relevant to the target class accuracy at every round. Empirically, computing the mean of \( L'_i \) for each client \( j \) across all rounds it participates in works well at identifying the clients contributing most to the target class.

### B. Dropping Attack

The targeted dropping attack starts with the adversary performing the Client Identification procedure, by running Algorithm 1 during the first \( T_N \) rounds of the protocol, in which training happens as usual. The adversary identifies in expectation \( k_N \) clients contributing updates for the target class. After Client Identification is performed, the network adversary drops contributions from the \( k_N \) clients in every round in which they participate after round \( T_N \). As the adversary can expect an increase in identification accuracy by monitoring more rounds of the protocol, they can repeat the Client Identification procedure in each subsequent round, and, if necessary, update the list of selected clients. We assume that the adversary drops the identified clients’ updates, and the server simply updates the model using all received updates. If the Client Identification protocol is not completely accurate, the attacker might drop traffic from non-target clients but we found this has a minimal impact on the trained FL model.

### C. Attack Analysis

Algorithm 1 uses several parameters: the number of clients to identify \( k_N \) and the number of rounds to wait before identification is successful \( T_N \), which we analyze here.

1) \textit{How many clients to drop?}: In the FL protocol, \( m \) out of \( n \) clients are selected at random in each round. Setting the number of dropping clients \( k_N \) is mainly a tradeoff between maximally damaging accuracy on the target data and remaining stealthy by not significantly compromising the overall model performance. If \( k_N \) is too large, significant benign updates may be removed, preventing the model from achieving good accuracy, and potentially allowing the server to identify that an active adversary, rather than standard packet loss, is to blame for the dropped updates. If \( k_N \) is too small, however, not enough clients will be dropped to have a significant impact on the model. We consider a range of values \( k_N \leq k \), where \( k \) is the number of clients holding examples from the targeted class, and show the attack effectiveness for each.

2) \textit{How many rounds are needed to identify the clients?}: Here, we discuss how to set the number of rounds \( T_N \) for client identification for both plain and encrypted communication. When setting \( T_N \), if the value is set too small, then the adversary will not have enough observations to reliably identify the contributing clients. However, if set too large, the adversary will allow benign training too much time, resulting in high accuracy on the target population, making it more difficult to mount the attack later. Suppose the adversary wants to wait until all \( n \) clients have participated in the protocol at least once. This is a well-studied problem, known as the coupon collector’s [20] (our setting additionally considers batched arrivals). In this variation of the problem, the adversary must observe an expected number of \( cn \log(n)/m \) batches before observing each client, for a small constant \( c \). With values of \( n = 100 \), \( m = 10 \), roughly 46 rounds are necessary [21]. Having established a connection to the coupon collector’s problem, we will extend it to model the adversary in each setting:

**Plain communication \textsc{Comm Plain}.** The baseline case is where the adversary receives \textit{every} client’s updates. We carry out a modification of the coupon collector analysis to identify the setting of \( T_N \) where a batch of \( m \) out of \( n \) distinct clients participate in each round, and the goal is to identify a set of \( k_N \) of \( k \) target clients. The number of batches to wait for the \( i \)-th client from the \( k \) target clients is \( t_i = \frac{n}{m(k-i+1)} \) in expectation, as the probability that any target client which has been unobserved appears is \( \frac{m(k-i)}{n} \). By summing this expected batch count over \( i \) from 1 to \( k_N \) (using the linearity of expectation), we find that the expected number of batches to wait for the first \( k_N \) clients is \( \frac{n}{m} \left( H_{k_N} - H_{k_N-k_N} \right) \), where \( H_i \) is the \( i \)-th harmonic number (and using \( H_0 = 0 \) if \( k_N = k \)). As \( k \) increases, this value tends towards \( \approx \frac{n}{m} \ln(k(k-k_N)) \) rounds (when \( k = k_N \), we replace \( \ln(0) \) with 0). To use a setting that is common in our experiments, where \( n = 60 \), \( m = 10 \), \( k = 15 \), and \( k_N = 15 \), we expect to wait roughly \( \approx \frac{n}{m} \ln(k) = 6 \ln(15) \) rounds, or roughly 16 rounds. Note that waiting for fewer clients \( k_N \) requires fewer rounds.

**Encrypted communication \textsc{Comm ENC} and \textsc{Comm ENC Limited}.** In the realistic case of encrypted communication, the adversary only has access to mean updates and cannot localize an improved target accuracy to a particular client. However, clients who repeatedly participate in rounds where the target accuracy improves, can be considered as more likely targets. Moreover, clients who participate in any round where the target accuracy does not improve can be identified as non-targets, and we analyze only rounds without target clients. An \( \alpha \) precision level at identification can be reached by removing \( n - k/\alpha \) clients which are known non-target clients. To analyze how many rounds it takes to collect \( n - k/\alpha \) non-target clients, we compute the probability that a batch will contain non-target clients, and then compute how many non-target batches are required to collect \( n - k/\alpha \) clients. To compute the probability that a batch contains non-target clients, we notice that there are \( \binom{n-k}{m} \) possible batches which have non-target clients, and there are a total of \( \binom{n}{m} \) batches which are selected uniformly at random. Then the probability a batch has non-target clients is \( \frac{n-k}{m} \approx (1 - \frac{k}{n})^m \).
accumulate $n - k/\alpha$ non-target clients, we can use comparable coupon collector analysis as before, making the observation that each non-target batch is guaranteed to have $m$ non-target clients. On average, an upper bound of $\frac{H_n - k}{\alpha} - H_k\alpha$ of batches is sufficient. As an example, in a setting we use in our experiments, $n = 60$, $k = 15$, and $\alpha = 10$, the probability that a batch contains non-target clients is roughly $5.6\%$. To reach a precision of $\alpha = 0.3$, we obtained a total of 26 batches on average. In practice, it is likely that precision can be even higher than this value due to overestimation from our analysis.

We note that our analysis is similar to analysis for the group testing problem [22], introduced by [23], used for pool testing. However, a key difference is that our algorithm must be capable of identifying members from noisy aggregate information, rather than the clean signal which is typically provided during group testing. It is possible that more sophisticated group testing algorithms can be used to improve Algorithm 1 further by overcoming this constraint.

D. Amplifying Dropping Attack with Model Poisoning

In order to amplify the effectiveness of the targeted dropping attack, the adversary may also collude with, or inject, malicious clients, whose presence in training is designed to further compromise the performance on the target distribution. Following Bagdasaryan et al. [4] and Sun et al. [7], we use a targeted model poisoning attack known as model replacement. Writing $\theta_*$ for the parameter of the global model $f_*$, in this attack, the adversary seeks to replace $\theta_*$ with a selected target $\theta_*$ (as is done in [4], [7], we use the $\theta_*$ resulting from a data poisoning attack on a compromised client’s local training procedure). The poisoned clients’ local training sets are sampled identically to clients with access to the target class, with the difference of changing the labels of target class samples to another, fixed class. The update that the adversary sends is $\theta_*$ (as is done in [4], [7]), we use the $\theta_*$ resulting from a data poisoning attack on a compromised client’s local training procedure). The poisoned clients’ local training sets are sampled identically to clients with access to the target class, with the difference of changing the labels of target class samples to another, fixed class. The update that the adversary sends is $\theta_*$ (as is done in [4], [7]), we use the $\theta_*$ resulting from a data poisoning attack on a compromised client’s local training procedure). The poisoned clients’ local training sets are sampled identically to clients with access to the target class, with the difference of changing the labels of target class samples to another, fixed class.

IV. DEFENSES AGAINST NETWORK-LEVEL ADVERSARIES

Several defenses against network-level attacks are known. For instance, defenses could monitor and detect faulty paths in the network [24], create resilient overlays [25], [26], or secure the routing protocols [27]. These defenses might increase robustness, but are generally not effective against stealthy attacks, such as targeted update dropping, or edge-level attacks, such as model poisoning. Often, the FL owner might not control the network, so we design FL-specific server defenses that complement existing network-level defenses against the attacks introduced in Section III.

Model Poisoning Countermeasures. A popular form of defense against data and model poisoning attacks in FL is to replace the Federated Averaging protocol with a secure aggregation scheme [28]–[31]. It was shown, however, that an adversary can evade this defense (e.g., [4]–[6]), and finding a secure aggregation method remains an open problem. Sun et al. [7] observed that model poisoning attacks with larger gradient norms are more effective, and therefore a natural defense is to reduce the norms. With this method, an update $g$ sent from a client is clipped to a maximum norm $C$ and becomes $\min\left(1, \frac{\|g\|}{C}\right)g$. Clipping works particularly well against model poisoning attacks in which the local client update is boosted by the attacker [4].

Algorithm 2: Server Defensive UpSampling Strategy

Data: Target Dataset $D^*_S$, rounds $T_S$, loss function $\ell$, client count $n$, count of clients to upsample $k_S$, upsample factor $\lambda$

Function UpSampling($D^*_S, \ell, k_S, T_S$):

1. $Z = \text{ClientIdentification}(D^*_S, \ell, T_S, k_S)$

2. $p = \frac{1 - \frac{k_S}{n}}{\frac{n - k_S}{n}}$ for $c \in [n]$

3. for $i \in Z$ do

   1. $p[i] = \lambda/n$

   2. // Increase sampling probability for highly contributors

4. return $p$

UpSampling Defense Strategy. While Clipping reduces the impact of model poisoning attack, we still need to defend against targeted update dropping. When a dropping attack is performed, the number of clients contributing legitimate updates to the target class is reduced, leading to a larger impact on naturally under-represented classes (i.e., available in a small set of clients). Thus, it is essential for the server to first identify clients contributing to the target class, and second, leverage them to improve the accuracy of the target population. For the first component, we can use directly Algorithm 1 for Client Identification, using the knowledge available to the server. In standard FL implementations, servers receive individual model updates from the clients, while in privacy-preserving FL implementations based on MPC [19] servers only receive aggregated updates. For the second task, the server can run Algorithm 2 (UpSampling defense) to modify the client sampling strategy in the FL protocol. With this modification, the sampling probability of identified clients is increased by a factor of $\lambda$ and the sampling probability of all others is decreased.

The server will determine how many clients to identify, $k_S$, and how many rounds to monitor, $T_S$, using its own target dataset $D^*_S$ to estimate contributions. As with the attacker, the server will repeat the Client Identification process at each successive round to refine its list of clients to up-sample. Interestingly, the UpSampling strategy can help even if there is no dropping attack, but there are simply too few clients from some target population on which the model accuracy is low.

V. EXPERIMENTAL EVALUATION

Here, we evaluate our attacks, show how model poisoning amplifies targeted update dropping damage, and conclude by looking at the performance of the UpSampling defense.

A. Experiment Setup

We use three well known datasets (EMNIST, FashionMNIST, and DBPedia-14), which we adapt to the non i.i.d. setting by controlling the class distribution across clients. In all cases, the
target population is represented by samples of an entire class selected from the dataset. Our datasets have balanced classes, and we use classes 0, 1, 9 for our evaluation. All reported results are averages of 4 trials, with different randomness seeds.

The list of parameters used in our FL protocol is shown in Table I. EMNIST [32] is a handwritten digits recognition dataset with 344K samples. We use approximately 100K images of numbers, partitioned equally, without overlap, among 100 clients. To enforce heterogeneity, we allocate samples from the target victim class to \( k \) fixed clients and vary \( k \). For those \( k \) clients, we allocate \( \alpha_T = 50\% \) of the local dataset to be target class points, while the remainder is sampled from the remaining classes according to a Dirichlet distribution with parameter \( \alpha_D = 1.0 \). For clients without the target class, we sample 100\% of the local training set from a Dirichlet distribution with \( \alpha_D = 1.0 \), following [33]. We train a convolutional model (two 2D convolution layers and two linear layers, with learning rate \( \eta_L = 0.1 \) for 100 rounds, selecting 10 clients at each round, using mean aggregation with a learning rate \( \eta = 0.25 \).

FashionMNIST [34] is an image classification dataset, consisting of 70K gray-scale images of 10 types of clothing articles. It is more complex than EMNIST, but smaller. Thus, we increase the number of training rounds to \( T \in \{200, 300\} \), reduce the number of clients from 100 to 60, and set \( \alpha_T \) to 0.6. We set the local dataset size to 400. We use a convolutional model similar to the one used before, and fix all other parameters.

DBPedia-14 [35] is an NLP text classification dataset consisting of 560K articles from 14 ontological categories, such as "Company", "Animal", "Film". DBPedia-14 is a relatively complex dataset, so we use the same \( k, T \), and \( \alpha_T \) as in FashionMNIST, and a local dataset size of 1000. The model is trained starting from pre-trained GloVe embeddings [36], followed by two 1D convolution layers and two linear layers, and optimized with Adam, with \( \eta_L = 0.001 \).

### B. Baselines: Perfect Knowledge and Random Dropping

To demonstrate the potential severity of a dropping attack, we evaluate the best possible case, where the adversary has perfect knowledge of a subset of \( k_N \) target clients from the beginning of the protocol, and drops every update originating from them throughout training. The results in Table II demonstrate the power of update dropping, and provide a baseline to compare our full attack pipeline against. We also evaluate the effect of an adversary that selects victim clients uniformly at random.

### C. Client Identification Evaluation

Table III shows the average number of targeted clients correctly identified by Algorithm 1 for encrypted communication with plain as comparison. Client Identification works very well for plain communication, and maintains a reasonable performance even when the adversary only sees aggregated updates (COMM_ENC). For instance, on DBPedia an average of 11.75 out of 15 clients are identified at 70 rounds under COMM_ENC. Interestingly, for FashionMNIST and DBPedia, it is easier to identify target clients than for EMNIST (where an average of 7 out of 15 clients are identified at round 70).

One hypothesis for this phenomenon is that classes are more distinct in complex datasets, leading the global model to forget the target class in rounds with no participating target clients, resulting in significant loss increases for those rounds.

We also use Table III to select parameters for the targeted dropping attack and validate our analysis from Section III-C. In the COMM_ENC scenario, we see that more rounds are necessary for successful identification, as expected from Section III-C. To identify between \( k/3 \) and \( 2k/3 \) of the target clients, we need to wait between 30 and 50 rounds, and, in many cases, the identification accuracy tends to plateau in successive...
TABLE IV
TARGETED DROPPING ATTACK, UNDER COMMPLAIN AND COMMENC.
ACCURACY ON TARGET POPULATION AT ROUNDS T/2 AND T, T = 100
FOR EMNIST, T = 200 FOR FASHIONMNIST AND DBPEDIA.

| Dataset | k   | Number of Clients Dropped kN |
|---------|-----|-----------------------------|
|         | 0   | 26/3                        |
| EMNIST  | 9   | 0.470/0.66 0.250/0.49 0.090/0.18 0.000/0.00 |
|         | 12  | 0.580/0.78 0.390/0.65 0.190/0.33 0.000/0.00 |
|         | 15  | 0.650/0.80 0.500/0.74 0.260/0.50 0.020/0.02 |
| FashionMNIST | 15  | 0.400/0.50 0.240/0.23 0.020/0.03 0.000/0.00 |
| DBPedia | 15  | 0.360/0.53 0.100/0.01 0.000/0.00 0.000/0.00 |
|         |     | COMMENC                     |
| EMNIST  | 9   | 0.470/0.66 0.350/0.38 0.320/0.52 0.320/0.52 |
|         | 12  | 0.580/0.78 0.500/0.72 0.480/0.69 0.450/0.67 |
|         | 15  | 0.650/0.80 0.630/0.78 0.600/0.76 0.560/0.71 |
| FashionMNIST | 15  | 0.400/0.50 0.340/0.38 0.140/0.13 0.030/0.01 |
| DBPedia | 15  | 0.360/0.53 0.190/0.06 0.060/0.00 0.000/0.00 |

rounds. Thus, we select round 30 as the starting point for the dropping attack in our COMMENC experiments.

D. Targeted Dropping Evaluation
We measure our targeted dropping attack’s performance in Table IV. Under COMMPLAIN, it significantly compromise target population accuracy, closely approximating the perfect knowledge adversary for increasing values of kN. We also observe that our attack in the COMMPLAIN scenario vastly outperforms the strategy of randomly dropping the same number of clients, in all situations. Our attacks’ performance decreases when the adversary’s knowledge is limited in COMMENC, which is expected from the reduced Client Identification performance. We still observe a significant advantage in using our identification pipeline, over the random selection baseline. For instance, with kN = 5, the target accuracy on DBPedia drops drastically from 53% to 6%. Moreover, these results highlight the trend mentioned in Section V-C: on more complex tasks, such as FashionMNIST and DBPedia, the high identification accuracy leads to noticeably larger attack performance, than in EMNIST. Given the targeted nature of our attack, in all considered scenarios, dropping the victim clients generally leads to very little degradation of the overall model performance – on average 3.88% for COMMPLAIN and 1% for COMMENC.

E. Impact of Model Poisoning and Targeted Dropping
We compared the effects of the model poisoning strategy and targeted dropping on the EMNIST dataset for the cases of perfect knowledge, COMMPLAIN, and COMMENC in Figures 1a, 1c, and 1e respectively. These heatmaps show that, for different levels of kN (number of dropped clients) and kP (number of poisoned clients), the model replacement attack is more effective than targeted dropping, even when the adversary has perfect knowledge. The results, however, are significantly different when Clipping, a standard poisoning defense [7], [37], [38], is applied to local model updates. Figures 1b, 1d, and 1f show the same set of experiments repeated with a clipping norm of 1. These heatmaps highlight that, while clipping lowers the impact of model poisoning, the combination of targeted dropping and model poisoning still results in very noticeable targeted performance degradation. For instance, under COMMPLAIN, the original model accuracy on class 0 is 0.8. This becomes 0.59 when dropping 7 clients out of 15, and 0.66 when poisoning 7 clients, while combining both attacks leads to 0.4 accuracy.

F. Impact of Adversarial Visibility
We model a targeted, resource-limited, adversary, COMMENC_LIMTED, by restricting their ability to observe the clients participating to the protocol to a fixed-size subset of the clients, chosen before the attack starts, sampled from a Dirichlet distribution with concentration parameter αV. αV controls how likely it is for the sampled visible set to include clients containing data of the target population, with larger values of αV leading to higher likelihood. We run the attack on FashionMNIST with a similar setup as in Table IV under the COMMENC conditions. The results reported in Figure 2a show that, as expected, higher visibility fractions, and larger αV lead to smaller target accuracy values. The network adversary is still quite effective even when observing a small fraction of the clients, for instance achieving a ≈ 43.6% relative target accuracy decrease when observing 20 out of 60 clients with αV = 2. Similarly, Figure 2b shows that adding poisoning always leads to better accuracy degradation.
G. Defense Evaluation

In Table V, we evaluate the defense strategies under COMM.PLAIN and COMM. ENC, using targeted dropping and poisoning attacks. We set the number of dropped ($k_N$) and poisoned clients ($k_P$) to $2k/3$ for EMNIST, and $k/3$ for FashionMNIST and DBPedia. These parameters result in strong attacks, as the target accuracy is below 4% under targeted dropping and poisoning, when no mitigation is used. UpSampling is successful in mitigating targeted dropping, and, when combined with Clipping, achieves high accuracy against the powerful combined attack. For instance, on EMNIST’s class 1 with $k = 9$, the update dropping attack causes accuracy to decrease from 92% to 25% under COMM.PLAIN, and UpSampling manages to restore the target accuracy to 76%.

The UpSampling defense is very effective under the COMM. ENC setting, due to knowledge asymmetry, i.e. the attacker receives aggregated updates, while the defender observes individual updates. Thus, UpSampling, defending against the dropping attack, improves the target model accuracy by an average (over the 3 classes) of 6.33% on EMNIST, 10.66% on FashionMNIST, and 24.66% on DBPedia for $k = 15$ compared to the original accuracy. Even under both targeted dropping and model poisoning attacks, the combined UpSampling defense and Clipping results in an average decrease of 5% on EMNIST and average increase of 9% on FashionMNIST and 16.66% on DBPedia over the 3 classes compared to the original accuracy. Interestingly, classes with lower original accuracy benefit more from the UpSampling strategy, with improvements as high as 34% (on DBPedia for class 9, original accuracy is increased from 60% to 94% with UpSampling under targeted dropping).

We observed that under-represented classes (in terms of number of clients holding samples from those classes) are impacted more by our attacks. To alleviate this problem, the server could identify new clients with data from the populations of interest, and add them to the set of clients participating in the FL protocol. In cross-device FL settings, servers typically have access to a large number of clients, and can make decisions on expanding the set of participating clients to improve accuracy on under-represented populations. We have also evaluated the attacks and defenses against FL protocols using secure aggregation [19].

Here, both the network-level adversary and the server only observe aggregated model updates. Even in this challenging setting, the server can identify clients contributing mostly to the target class, making UpSampling combined with Clipping quite effective. Results are omitted for space limitation.

VI. RELATED WORK

Given the distributed training process in FL, poisoning attacks represent an even larger threat than in traditional ML systems. For instance, poisoning availability attacks have been shown effective in Federated Learning in recent work [5], [6]. Targeted model poisoning attacks impact a small population, or introduce a backdoor in the models to mis-classify instances containing the trigger [3], [4], [7], [8]. Researchers also proposed methods to defend the FL protocol from adversaries, such as Byzantine-resilient or trust-based aggregation rules [28]–[31], [39]–[43]. However, [5] and [6] systematically analyzed Byzantine-robust aggregation schemes, showing that an adversary controlling compromised clients can bypass these defenses with poisoning availability attacks. Also, targeted poisoning attacks can bypass Byzantine-resilient aggregation, such as Krum [4]. Methods protecting from model poisoning include filtering of malicious gradients for availability attacks [5], [6], [9] and gradient clipping for targeted attacks [4], [7].

VII. CONCLUSION

We examined the effects that a network-level adversary can have on the final model’s accuracy on a target population in cross-device FL. We proposed a new attack, based on a procedure for identifying the set of clients mostly contributing towards a target class of interest to the adversary, and showed that it can be amplified by coordinated model poisoning attacks. We showed that our attacks are effective for realistic scenarios where the communication is encrypted and the attacker has limited network visibility. We also explored defensive approaches, and found that our UpSampling mechanism can be extremely successful when paired with clipping of model updates, against very powerful network-level attacks.

ACKNOWLEDGEMENTS

This research was supported by the Department of Defense Multidisciplinary Research Program of the University Research Initiative (MURI) under contract W911NF-21-1-0322.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in AISTATS. PMLR, 2017.
[2] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, “Data Poisoning Attacks Against Federated Learning Systems,” arXiv:2007.08432, 2020.
[3] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, “Model Poisoning Attacks in Federated Learning,” in NeurIPS SECML, 2018.
[4] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How to backdoor federated learning,” in AISTATS. PMLR, 2020.
[5] M. Fang, X. Cao, J. Jia, and N. Z. Gong, “Local Model Poisoning Attacks to Byzantine-Robust Federated Learning,” in USENIX Security, 2020.
[6] V. Shejwalkar and A. Houmansadr, “Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning,” in NDSS, 2021.
[7] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, “Can you really backdoor federated learning?” arXiv:1911.07963, 2019.
[8] H. Wang, K. Sreenivasan, S. Rajpoot, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, “Attack of the Tails: Yes, You Really Can Backdoor Federated Learning,” in NeurIPS, 2020.

2Additional results are presented in https://arxiv.org/abs/2208.12911
| $k$ | Target Class | No Attack | FedAvg | Targeted Drop | Poisson Attack | Targeted Drop + Poison | DBPedia |
|-----|--------------|-----------|--------|---------------|----------------|-------------------------|---------|
|     |              |           |        | COMM_PLAN | COMM_ENC |           |           |        | COMM_PLAN | COMM_ENC |           |           |         |
|     |              |           |        |           |          |           |           |         |           |          |           |           |         |
|     |              |           |        | Clip | UpS | Clip | UpS | Clip | UpS | Clip | UpS | Clip | UpS | Clip | UpS | Clip | UpS |
|     |              |           |        |           |          |           |           |         |           |          |           |           |         |
| 0   | 0.66         | 0.18      | 0.44   | 0.52     | 0.76     | 0.30      | 0.64      | 0.04  | 0.04  | 0.00  | 0.00  | 0.31  | 0.31  | 0.00  | 0.04  | 0.04  | 0.68  |
| 1   | 0.92         | 0.25      | 0.76   | 0.76     | 0.96     | 0.30      | 0.04      | 0.04  | 0.00  | 0.00  | 0.00  | 0.30  | 0.30  | 0.00  | 0.04  | 0.04  | 0.88  |
| 9   | 0.36         | 0.17      | 0.40   | 0.46     | 0.70     | 0.23      | 0.01      | 0.01  | 0.00  | 0.00  | 0.00  | 0.23  | 0.23  | 0.00  | 0.05  | 0.05  | 0.57  |
| 0   | 0.78         | 0.33      | 0.66   | 0.69     | 0.85     | 0.40      | 0.09      | 0.09  | 0.00  | 0.00  | 0.00  | 0.40  | 0.40  | 0.00  | 0.36  | 0.36  | 0.77  |
| 1   | 0.95         | 0.55      | 0.85   | 0.91     | 0.97     | 0.60      | 0.04      | 0.04  | 0.00  | 0.00  | 0.00  | 0.60  | 0.60  | 0.00  | 0.43  | 0.43  | 0.92  |
| 9   | 0.67         | 0.31      | 0.52   | 0.56     | 0.78     | 0.29      | 0.02      | 0.02  | 0.00  | 0.00  | 0.00  | 0.29  | 0.29  | 0.01  | 0.27  | 0.27  | 0.67  |
| 0   | 0.80         | 0.50      | 0.71   | 0.76     | 0.89     | 0.35      | 0.06      | 0.06  | 0.00  | 0.00  | 0.00  | 0.35  | 0.35  | 0.00  | 0.44  | 0.44  | 0.71  |
| 15  | 0.96         | 0.79      | 0.94   | 0.94     | 0.97     | 0.51      | 0.18      | 0.18  | 0.00  | 0.00  | 0.00  | 0.51  | 0.51  | 0.00  | 0.48  | 0.48  | 0.94  |
| 9   | 0.75         | 0.39      | 0.59   | 0.58     | 0.85     | 0.40      | 0.03      | 0.03  | 0.00  | 0.00  | 0.00  | 0.40  | 0.40  | 0.01  | 0.35  | 0.35  | 0.71  |

|     |              |           |        |           |          |           |           |         |           |          |           |           |         |
|     |              |           |        |           |          |           |           |         |           |          |           |           |         |
| 0   | 0.47         | 0.36      | 0.62   | 0.39     | 0.74     | 0.58      | 0.38      | 0.38  | 0.19  | 0.58     | 0.02  | 0.36  | 0.17  | 0.69  |
| 1   | 0.95         | 0.93      | 0.96   | 0.94     | 0.96     | 0.93      | 0.77      | 0.77  | 0.28  | 0.93     | 0.03  | 0.81  | 0.24  | 0.95  |
| 9   | 0.87         | 0.81      | 0.93   | 0.82     | 0.94     | 0.89      | 0.62      | 0.62  | 0.19  | 0.89     | 0.05  | 0.59  | 0.23  | 0.92  |
| 15  | 0.54         | 0.10      | 0.82   | 0.12     | 0.84     | 0.77      | 0.00      | 0.00  | 0.01  | 0.77     | 0.02  | 0.00  | 0.01  | 0.79  |
| 9   | 0.60         | 0.17      | 0.91   | 0.12     | 0.94     | 0.62      | 0.00      | 0.00  | 0.02  | 0.62     | 0.00  | 0.00  | 0.00  | 0.76  |

**TABLE V**

| Target class accuracy, under COMM_PLAN and COMM_ENC at $T = 100$ for EMNIST, and $T = 300$ for FashionMNIST and DBPedia.

**Targeted Dropping and Targeted Dropping + Poisoning attack scenarios, and both Clipping (Clip) and UpSamping (UpS) defenses.**