Not One Less: Exploring Interplay between User Profiles and Items in Untargeted Attacks against Federated Recommendation

Yurong Hao
Beijing Jiaotong University
Beijing, China
yurong.hao@bjtu.edu.cn

Xihui Chen
University of Luxembourg
Luxembourg, Luxembourg
xihui.chen@uni.lu

Xiaoting Lyu
Beijing Jiaotong University
Beijing, China
xiaoting.lyu@bjtu.edu.cn

Jiqiang Liu
Beijing Jiaotong University
Beijing, China
jqliu@bjtu.edu.cn

Yongsheng Zhu
Beijing Jiaotong University
China Academy of Railway Sciences Corporation Limited
Beijing, China
19111078@bjtu.edu.cn

Zhiguo Wan
Zhejiang Lab
Hangzhou, China
wanzhiguo@zhejianglab.com

Sjouke Mauw
University of Luxembourg
Luxembourg, Luxembourg
sjouke.mauw@uni.lu

Wei Wang*
Beijing Jiaotong University
Beijing, China
wangwei1@bjtu.edu.cn

ABSTRACT

Federated recommendation (FR) is a decentralised approach to training personalised recommender systems, protecting users' privacy by avoiding data collection. Despite its privacy advantages, FR remains vulnerable to poisoning attacks. We focus on untargeted poisoning attacks against FR which degrade the overall performance of recommender services, leading to a detrimental impact on user experience and service quality. In this paper, we propose a general framework to formalise untargeted attacks and identify the vital role played by the interplay between items and user profiles in determining FR's performance. We present an untargeted attack `FRecAttack`^{2} which exploits this interplay. Specifically, we develop various methods for sampling user profiles, which approximate user distributions with and without collusion among malicious users. Then we leverage a new measurement to identify items that can disrupt the original interplay with user profiles, based on the change velocity of items' recommendation scores during optimisation. Extensive experiments demonstrate the superiority of our attack, outperforming existing methods by up to 27.56%, and its stealthiness in evading mainstream defences. To counteract untargeted attacks, we present a defence `GuardCQ` to detect malicious users by quantifying their contribution to boost the right interplay between items and user profiles. Empirical results show that `GuardCQ` effectively mitigates the attack's impact on FR and enhances the robustness of FR against poisoning attacks.

CCS CONCEPTS

• **Security and privacy** → **Distributed systems security**; • **Information systems** → *Information systems applications.*

KEYWORDS

Federated Recommendation, Model Poisoning, Untargeted Attack

ACM Reference Format:
Yurong Hao, Xihui Chen, Xiaoting Lyu, Jiqiang Liu, Yongsheng Zhu, Zhiguo Wan, Sjouke Mauw, and Wei Wang. 2024. Not One Less: Exploring Interplay between User Profiles and Items in Untargeted Attacks against Federated Recommendation. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3670365

1 INTRODUCTION

Personalised recommendations have become an integral feature of modern applications, ranging from news feeds and streaming services to e-commerce platforms. These systems aim to select and provide valuable information to users based on their preferences and interests. In general, a recommender system learns the characteristics of user profiles and items from past user-item interactions and other accessible information, enabling it to recommend items that are most matched to users' profiles. Traditionally, a central server, which has access to all data, is responsible for learning the model and running recommendation services. However, the rise in public awareness of privacy concerns in recent years requires decentralised solutions to address the risk of data misuse and privacy breaches at the single data curator. Federated learning (FL) enables collaborative model training without exposing users' data [4, 22]. In each training round, randomly selected users perform local training

*Corresponding Author: Wei Wang

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA.

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3670365

of a recommendation model and share their parameter gradients with an aggregator, which then combines all the gradients to update the global model parameters. The integration of FL with recommendation gives rise to decentralised recommender systems, commonly known as federated recommendation (FR) [1]. Webank stands as a typical real-world application of FR. They achieved a remarkable 30% increase in the discount rates for financial product recommendations by integrating FR into the banking enterprise.

To protect the sensitive nature of users' profiles, two privacy protection approaches have been explored in the literature within the context of FR, namely "*secret-hiding*" and "*DP-based*" methods. Unlike FL, where users share gradient updates for all parameters, secret-hiding FR requires users to update parameters locally concerning their profiles without sharing them with the aggregator [17, 18]. On the other hand, the DP-based approach employs differential privacy (DP) [9] by introducing carefully crafted noise to perturb all parameter gradients, offering a more robust privacy protection mechanism. The DP-based approach facilitates the sharing of noisy user profiles among users, allowing for the training of complex deep learning models such as graph neural networks (GNN) [11] requiring access to neighbouring users' profiles [19, 31].

Derived from FL, FR is vulnerable to poisoning attacks when certain users are controlled by adversaries [4]. According to the attack goals, it can be categorised into *targeted attacks*, aiming to promote specific items for economic gain, and *untargeted attacks*, aiming to degrade the overall recommendation performance. Unlike targeted attacks [25, 26, 40], untargeted attacks can inflict severe damage on nearly every user's experience, even invisibly reshaping individual judgement criteria for the right values and further changing their real-world decision-making, yet they have not received adequate attention in research. Our objective of this paper is thus to *rigorously* examine untargeted attacks on FR and unveil their true potential for damage, which has been underestimated in prior research.

To achieve this goal, we start by proposing a formal framework to generalise the recent advances of FR which allows us to further formalise untargeted attacks including their objectives and essential building blocks. The framework also unveils one significant limit of prior attacks, that is, the key role of the interplay between general users' profiles and items in determining the performance of FR systems. The prevailing concept behind state-of-the-art untargeted attacks is to deceive the optimisation process by promoting unpopular items while downgrading the recommendation ranks of popular items based on malicious users' profiles. For example, FedAttack [32] adopts hard sampling during each training round to identify the least possible positive interactions (termed hardest positive items) and the least possible negative ones (known as hardest negative items) for malicious users, optimising the model parameters to recommend these hard positive items. Prior attacks fail to capture the essence of untargeted attacks which are to degrade the recommendation performance of *general users*, not just malicious users. Independent parameter optimisation tailored to individual malicious users' profiles will limit the overall attack performance. A direct consequence is that a larger proportion of malicious users are required to simulate the distribution of general users accurately.

To address this limitation, we introduce a novel untargeted attack by enhancing existing attack methodologies with two essential building blocks: user profile sampling and interaction sampling. Our approach leverages user profile sampling to simulate the distribution of general users. Departing from the previous works' assumption of a global Gaussian distribution [25], we employ a series of finer-granular sampling methods, incorporating available information such as malicious users' profiles and attackers' power in coordinating communication among malicious users – an aspect that has not been previously explored. Using the sampled virtual user profiles, we calculate their hardest positive and negative item interactions. Traditional attacks rely solely on absolute recommendation scores for interaction sampling, which we validate as effective only in the later stages of model training but suboptimal in the early stages. Consequently, valuable time is wasted during the early stages, increasing the risk of detection by the aggregator due to the prolonged attack duration. To address this, we propose a solution that ensures an accurate early prediction of item popularity by complementing absolute scores with the velocity of recommendation score changes during training. In contrast to existing attacks that solely target secret-hiding FR models [1, 6, 24], our novel approach is versatile enough to be applicable to DP-based FR models as well. Another notable advantage of our method is that it does not rely on any prior knowledge that is not available to malicious users. By eschewing the need for undisclosed information, our attack ensures a more realistic and practical scenario for evaluation and analysis. We further propose a potential countermeasure `GuardCQ` based on contribution quantification. Specifically, the aggregator is required to track and record the users' rating behaviours and further detect malicious users by measuring their contributions to the right interplay between items and user profiles based on their behaviour. Our `GuardCQ` defence proves highly effective in mitigating the negative impact and can integrate with many existing Byzantine-robust FL methods to improve the robustness of FR against untargeted poisoning attacks. In summary, we have the following contributions:

- We present a versatile framework capable of formalising not only secret-hiding models but also recent DP-based models. Our framework provides a systematic way to formally define untargeted attacks against FR systems, revealing the underexplored interplay between general user profiles and items in prior research.
- We propose and develop a novel untargeted attack that thoroughly explores the interactions between user profiles and items. Our innovative approach also considers attack scenarios where malicious users collaborate under the coordination of attackers, making it more comprehensive and realistic.
- We propose a new defence to detect malicious users by quantifying their contributions to the right interplay between items and user profiles. Our defence can work in conjunction with many existing Byzantine-robust FL methods and provide more robust protection for the FR system.
- We conduct extensive experiments on four real-life datasets with both secret-hiding and DP-based FR systems to demonstrate the effectiveness of our `FRecAttack`^{2} and `GuardCQ`.

2 RELATED WORK

Attack for FL. FL was first proposed by McMahan et al. [14], with the primary objective to address inherent challenges regarding data privacy and centralisation in conventional machine learning

approaches. Due to the protection of users' data privacy, FL has been deployed by high-profile companies such as Google and Apple.

Unfortunately, FL is vulnerable to adversarial behaviours of malicious users [5, 15, 21, 23, 27] aiming to poison model optimisation during training. Based on attack goals, we can have two categories: *targeted* and *untargeted* attacks. Targeted attacks [21, 27] aim to minimise the accuracy of *specific* test inputs while ensuring that the global model performance remains unaffected while untargeted attacks [2, 10] minimise the accuracy for *all* test input thereby compromising the global model performance. Due to the broad impacts on the availability and integrity of the model, untargeted attacks are usually considered more detrimental to FL. Another classification way is based on the attacker's capabilities, which also divides existing attacks into two categories: *data* and *gradient* poisoning attacks. Data poisoning attacks [13, 28] *indirectly* alter model stochastic gradient descents(SGD) by manipulating training samples of malicious users, such as flipping training samples' labels [29]. In gradient poisoning attacks [2, 3, 7, 16], attackers *directly* change the model gradients to interfere with the model's optimisation process.

Attack for FR. FR is a special application domain of FL where parameters concerning user profiles require special protection. According to the protection mechanism, we first classify FR models into two classes: *secret-hiding* and *DP-based*. In secret-hiding models [6, 24], user embeddings are kept locally on each client while DP-based models [19, 31] perturb model parameters with noises satisfying differential privacy [9]. The introduction of protection mechanisms makes many attacks against FL ineffective. Furthermore, the extensive number of clients in the FR system also raises the difficulty and cost of attacks. Previous attacks may become significantly diluted or even rendered ineffective. Thus, specific attacks against FR are proposed consecutively [25, 26, 32, 37, 40]. PipAttack[40] and A-hum [25] have devoted efforts to promote target items by increasing their chances of being exposed to as many users as possible. These targeted attacks are often driven by economic incentives. Wu et al. [32] proposed the first untargeted attack named FedAttack to poison item embedding gradients to favour the hardest negative samples. Subsequently, Yu et al. [37] proposed ClusterAttack, which converges all item embeddings in FR into several dense clusters to make FR generate similar scores for these close items in the same cluster, and thus mess up the ranking order in the recommendation. However, these attacks only manipulate item embeddings to degrade the model performance and ignore the fact that it is the interplay of items and user profiles that determines the performance of FR. Moreover, the manipulation made to all item embeddings such as in ClusterAttack is easily detected by existing anomaly detectors due to the FR's inherent sparsity of user-item interactions. As a result, we aim to capture the essence of untargeted attacks and leverage the interplay of items and user profiles to construct a new attack to unveil the true risk exposed to FR models. Meanwhile, we propose a new defence to strengthen the robustness of FR models because it is still under-explored.

3 FORMALISING FR AND UNTARGETED ATTACKS

As we mentioned previously, compared to targeted attacks on FR, untargeted attacks have not been formally analysed in terms of attack objectives and attacking strategies of the adversary. In this section, we will present the first framework to generally formalise untargeted attacks and incorporate recent advances. Benefiting from this framework, we manage to identify and summarise the challenges which have not been well addressed in prior research.

3.1 Formalising FR systems

Let $\mathcal{I}$ and $\mathcal{U}$ denote the set of items and the set of users/devices, respectively. Each user $u \in \mathcal{U}$ stores a set $\mathcal{D}_u$ recording his/her past interactions with items in $\mathcal{I}$ of the form of $(u, \eta, r_{u,\eta})$ for each $\eta \in \mathcal{I}$. The value $r_{u,\eta}$ indicates whether user u ever had an interaction with η, e.g., whether a user watched a movie on NetFlix. In some cases, $r_{u,\eta}$ also infers the level or intensity of the interaction. For instance, users may rate a movie with a number of stars to show how much they like the movie. In this paper, we follow the general settings adopted in prior works about FR [12, 25] and adopt binary ratings, i.e., $r_{u,\eta} \in \{0, 1\}$. If $r_{u,\eta} = 1$, then a *positive interaction* occurred between user u and item η. Otherwise, no interaction exists yet between u and η and it is considered as a *negative interaction*. For scenarios where ratings are not binary, thresholds can be introduced to determine the positive interactions and label all interactions with the rest items as negative. Users store their past item interactions locally and keep them secret from others to protect their privacy.

When training a FR model, users collaborate with each other without sharing their past interactions. To coordinate the training, an aggregator is deployed to aggregate users' local updates in each round, and distribute the aggregated ones to users before the next round. Finally, three components are learned: the parameter values of a global recommendation function ψ denoted by $\mathbf{\Theta}_\psi$, the user embedding matrix $\mathbf{X}$ and the item embedding matrix $\mathbf{Y}$. We use $\mathbf{x}_u$ to represent the corresponding row of user u in $\mathbf{X}$ which represents the latent embedding vector of user u. Similarly, $\mathbf{y}_\eta$ represents the latent vector of item η. The recommendation function ψ parameterised by $\mathbf{\Theta}_\psi$ takes as input the embeddings of a user u and an item η, and outputs a score in $[0, 1]$ indicating the extent to which the item matches the user's profile, i.e., $\psi(\mathbf{x}_u, \mathbf{y}_\eta) \in [0, 1]$. Intuitively, we can interpret $\mathbf{x}_u$ and $\mathbf{y}_\eta$ as summaries of the features of user u and item η which are the most effective to determine whether u and η will interact in the near future under function ψ. The K items with the largest scores which user u has not interacted with compose the list of recommended items, denoted by $\mathcal{R}_u = Top_K(\mathbf{x}_u, \mathbf{Y}, \psi)$.

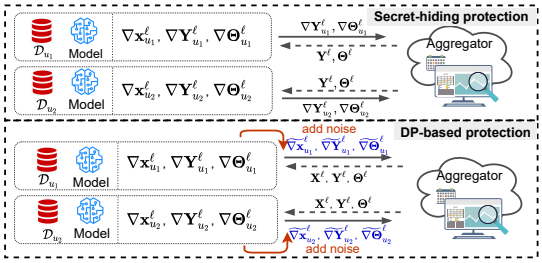

Figure 1: User privacy protection in FR.

As exposing user profiles may breach users' privacy, a user's true profile is only accessible to the owner. This holds during the training and also when a well-trained model is applied to make

recommendations for users. As a result, users can actually curate more parameters than the aggregator. We introduce $\mathcal{P}_{agg}$ and $\mathcal{P}_{user}$ to denote the set of parameters maintained by the aggregator and users, respectively. In practice, the settings of $\mathcal{P}_{agg}$ and $\mathcal{P}_{user}$ depend on the type of FR models, i.e., secret-hiding or DP-based. In Figure 1, we illustrate the general data flow between the aggregator and users in the models of these two categories with two users and round ℓ as an example. Recall that secret-hiding models protect users' private data by forcing users to process the parameters concerning their profiles locally and keep the SGD updates secret from the aggregator while DP-based methods explore the concept of differential privacy and perturb SGD updates with crafted noises to control the amount of information leakage. We can see that in secret-hiding models, $\mathcal{P}_{agg}$ only consists of the parameters of the item embedding matrix and the parameters of the recommendation function, i.e., $\mathbf{Y}$ and $\mathbf{\Theta}_\psi$, thus we have $\mathcal{P}_{agg} \subset \mathcal{P}_{user}$. Due to the perturbation mechanism, DP-based models allow users to share all parameters including those of their user profiles. Therefore, $\mathcal{P}_{agg}$ and $\mathcal{P}_{user}$ are the same and consist of $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{\Theta}_\psi$.

In each round ℓ, the aggregator selects randomly a set of users in $\mathcal{U}$ and sends them the values of the parameters in $\mathcal{P}_{agg}$ updated in the previous round. With the parameter values received from the aggregator, each user optimises the local model and calculates the SGDs for parameters in $\mathcal{P}_{user}$. Users send back their local SGDs of the parameters $\mathcal{P}_{agg}$ to the aggregator and update the sensitive parameters that are not curated by the aggregator. Then the aggregator aggregates received SGDs from all selected users and updates the parameters in $\mathcal{P}_{agg}$ for the next round $\ell+1$. The training terminates when the maximum pre-defined number of rounds, i.e., T, is reached. Let $\overline{\mathcal{R}}_u \subseteq \mathcal{I}$ be the items that perfectly match the user u's profile and should be recommended. Then the goal of FR model training can be interpreted as calculating the optimal gradients in maximum T rounds to minimise the average difference between the recommendations made by the trained system for every user u and the ideal recommendation item set $\overline{\mathcal{R}}_u$. Let $Error(\mathcal{R}_u, \overline{\mathcal{R}}_u)$ be a measurement quantifying the difference between the two sets. For instance, for recommender systems, it can be determined as the exposure rate, i.e., $\frac{|\overline{\mathcal{R}}_u/\mathcal{R}_u|}{|\overline{\mathcal{R}}_u|}$. Recall $\mathcal{R}_u = Top_K(\mathbf{x}_u^T, \mathbf{Y}^T, \psi^T)$. Formally, the objective of FR training can be described as follows:

$$\arg \min_{\nabla \mathcal{P}_{user}^{u,\ell}:\ell=1,\ldots,T \wedge u \in \mathcal{U}} \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} Error(\mathcal{R}_u, \overline{\mathcal{R}}_u).$$

3.2 Untargeted attacks against FR

In the attacks on FR, the adversary controls a set of malicious users denoted by $\widehat{\mathcal{U}}$. These malicious users are normal users of the FR system with a set of past item interactions, thus $\widehat{\mathcal{U}} \subset \mathcal{U}$. Note that due to the large number of users of recommender systems, the adversary rarely has the power to control all users. These malicious users can modify their data and manipulate the SGD updates in each round. The objective of the adversary is to collaborate with malicious users to disrupt the aggregation of SGD updates at the aggregator. Let $\nabla \widehat{\mathcal{P}}_{agg}^{u,\ell}$ be the gradient updates from user $u \in \widehat{\mathcal{U}}$ at round ℓ. Intuitively, the attack goal is to calculate the best gradient updates from all malicious users that can deceive the aggregator and other benign users to miscalculate model parameters that can yield recommendations with the maximum error from the true recommendations. Let $\widehat{\mathbf{x}}_u^T$, $\widehat{\mathbf{Y}}^T$ and $\widehat{\psi}^T$ be the final user profile of a user $u \in \widehat{\mathcal{U}}$, item embeddings and recommendation function trained with the pre-defined maximum rounds under the untargeted attack. With $\widehat{\mathcal{R}}_u = Top_K(\widehat{\mathbf{x}}_u^T, \widehat{\mathbf{Y}}^T, \widehat{\psi}^T)$, the ultimate goal of an untargeted attack can be formalised as follows:

$$\arg \max_{\nabla \widehat{\mathcal{P}}_{agg}^{u,\ell}:\ell=1,\ldots,T \wedge u \in \widehat{\mathcal{U}}} \frac{1}{\widehat{\mathcal{U}}|\mathcal{U}/\widehat{\mathcal{U}}|} \sum_{u \in \mathcal{U}/\widehat{\mathcal{U}}} Error(\mathcal{R}_u, \widehat{\mathcal{R}}_u).$$

In addition to attack performance measuring the recommendation quality, a successful untargeted attack should also be stealthy and covert so that it is difficult to be detected by deployed defences. In practice, as discussed in [38], it is intractable to calculate the optimal combination of the gradient updates of all rounds. As a result, untargeted attacks then take a greedy approach to maximise the error rates in each round independently. This can be formally described as follows. For each round ℓ,

$$\arg \max_{\nabla \widehat{\mathcal{P}}_{agg}^{u,\ell}:u \in \widehat{\mathcal{U}}} \frac{1}{\widehat{\mathcal{U}}|\mathcal{U}/\widehat{\mathcal{U}}|} \sum_{u' \in \mathcal{U}/\widehat{\mathcal{U}}} Error(\mathcal{R}_{u'}, \widehat{\mathcal{R}}_{u'}^{\ell+1})$$

where $\widehat{\mathcal{R}}_u^{\ell+1} = Top_K(\widehat{\mathbf{x}}_u^\ell, \widehat{\mathbf{Y}}^\ell, \widehat{\psi}^\ell)$. From the above equation, we can identify three challenges. First, the attack performance is an overall estimation of the recommendation error rates on all benign users. It is not quantified as effective if the attacker only manipulates the recommendations of a specific user or a user subgroup. Second, the recommendations made by the original system and the manipulated system under attack for any benign user $u \in \mathcal{U}/\widehat{\mathcal{U}}$, i.e., $\mathcal{R}_u$ and $\widehat{\mathcal{R}}_u^{\ell+1}$, are not accessible. This is because the normal training will never be executed due to the attack, and the accurate user embeddings are protected and stored locally at the users' sides. Third, the error functions based on the differences between two sets are difficult to optimise efficiently with methods such as backpropagation.

As a result, to implement an effective attack and address the above challenges, each malicious user needs to prepare two sets of virtual data. One is the user embeddings for a set of virtual users $\ddot{\mathcal{U}}^\ell$ at each round ℓ, i.e., $\mathbf{X}_{\ddot{\mathcal{U}}^\ell}$. The other is a carefully crafted item set for each user $\ddot{u} \in \ddot{\mathcal{U}}^\ell$ denoted by $\ddot{\mathcal{D}}_{\ddot{u}}^\ell$. These two sets are "virtual" in the sense that they do not have to correspond to any real user in $\mathcal{U}$. The virtual user embedding set is created with the purpose to represent samples drawn from the same distribution as benign users. The item interaction set is to mislead the training to increase the recommendation score of the most unfavourable items and decrease the preferable ones for any user. An ideal $\ddot{\mathcal{D}}_u^\ell$ for a benign user can be obtained by flipping the labels of u's past item interactions. However, due to the virtual nature of sampled users, we do not have access to their past interactions. Therefore, we need to use other methods to construct $\ddot{\mathcal{D}}_{\ddot{u}}$. The hard sampling method [33] is one plausible solution which has been adopted and shown effective in [32]. If a given user is very likely to interact with an item, then this item is called a *hard negative* sample. Otherwise, it is a *hard positive* sample. Then $\ddot{\mathcal{D}}_{\ddot{u}}^\ell$ is constructed by using the hardest negative samples as negative interactions, i.e., by assigning 0 as the ratings and the hardest positive samples as positive interactions by setting rating as 1. With $\ddot{\mathcal{U}}^\ell$ and $\ddot{\mathcal{D}}_{\ddot{u}}^\ell$ for each $\ddot{u} \in \ddot{\mathcal{U}}$, the goal of an untargeted attack at round ℓ is as follows:

Algorithm 1: Pseudocodes of a malicious user $\widehat{u}$.

for *round* $\ell = 1, \dots T$ **do**
  $VP^{\ell}_{agg} = \texttt{ReceiveFromAggregator}(\mathcal{P}_{agg})$
  for $(p, v^{\ell}_{p}) \in VP^{\ell}_{agg}$ **do**
    $VP^{\ell}_{user} = \texttt{UpdateParaValue}(VP^{\ell}_{user}, p, v^{\ell}_{p})$
  end
  $\ddot{\mathcal{U}}^{\ell} = \texttt{SampleUserEmbedding}(VP^{\ell}_{user})$
  $\ddot{\mathcal{D}}^{\ell}_{\ddot{\mathcal{U}}} = \texttt{SampleItemInteraction}(\ddot{\mathcal{U}}^{\ell}, VP^{\ell}_{user})$
  $\nabla\widehat{\mathcal{P}}^{\ell}_{agg} = \arg\min_{\nabla\widehat{\mathcal{P}}_{agg}} \texttt{Loss}(\mathbf{X}^{\ell}_{\ddot{\mathcal{U}}}, \ddot{\mathcal{D}}^{\ell}_{\ddot{\mathcal{U}}}, VP^{\ell}_{user})$
  $\texttt{UploadSGDToAggregator}(\nabla\widehat{\mathcal{P}}^{\ell}_{agg})$
end

$$\arg\min_{\nabla\widehat{\mathcal{P}}^{\widehat{u},\ell}_{agg}:\widehat{u}\in\widehat{\mathcal{U}}} \frac{1}{|\ddot{\mathcal{U}}|}\sum_{\ddot{u}\in\ddot{\mathcal{U}}^{\ell}} Loss_{attack}(\mathbf{x}^{\ell}_{\ddot{u}}, \mathbf{Y}^{\ell}, \ddot{\mathcal{D}}^{\ell}_{\ddot{u}}, \psi^{\ell})$$

where $Loss_{attack}(\mathbf{x}^{\ell}_{\ddot{u}}, \mathbf{Y}^{\ell}, \ddot{\mathcal{D}}^{\ell}_{\ddot{u}}, \psi^{\ell}) =$

$$\sum_{(\ddot{u},\eta,r)\in\ddot{\mathcal{D}}^{\ell}_{\ddot{u}}} \Big[r \log \psi(\mathbf{x}^{\ell}_{\ddot{u}}, \mathbf{y}^{\ell}_{\eta}) + (1-r)\log\big(1-\psi(\mathbf{x}^{\ell}_{\ddot{u}}, \mathbf{y}^{\ell}_{\eta})\big)\Big].$$

We adopt cross-entropy to formulate the loss function. Other loss functions can also be used without loss of generality. Alg. 1 summarises the main steps of an untargeted attack executed by a malicious user. In each round, the same as a benign user, a malicious user receives the parameters from the aggregator with which it updates its local parameters. Then the malicious user calculates a set of user samples to approximate the distribution of general benign users. With the sampled users, the user continues to calculate the hard positive samples and hard negative user-item interactions. In the end, with the interaction samples and user samples, the malicious user computes the optimal SGDs of the parameters that will be uploaded to the aggregator according to the loss function.

Discussion. Benefiting from this framework, we unveil the two components that are critical to determining the performance of untargeted attacks: *user sampling* and *interaction sampling*. The former determines whether the distribution of benign users can be simulated while the latter directly affects the misleading effects of the manipulated gradient updates from malicious users. These essential two components actually capture the important role played by the *interplay* between a general user's profile and items during the iterative process of the attack. A successful untargeted attack requires a close simulation of the distribution of benign users and an accurate inference of items that match given benign users' profiles and should be finally recommended.

3.3 Identified challenges

The untargeted attacks presented in the literature can actually be described by our framework. For instance, in FedAttack [32], each malicious user $\widehat{u}$ updates his/her user embedding vector $\mathbf{x}_{\widehat{u}}$ from local data through the normal training. Then the malicious user retrieves the hardest negative (or positive) samples with the closest (or farthest) distance to the $\mathbf{x}_{\widehat{u}}$ from the updated item embeddings stored in $\mathbf{Y}$. This can be viewed as an instance of our framework with $\ddot{\mathcal{U}} = \{\widehat{u}\}$ and $\mathbf{X}_{\ddot{\mathcal{U}}} = \mathbf{x}_{\widehat{u}}$. We identify the following challenges that have not been well addressed in prior works:

- Individual malicious users' representations are simply used as the representatives of benign users. One direct consequence is that a sufficiently large number of malicious users is required to ensure the set of malicious users can actually approximate the original distribution. Moreover, it is also not practical to assume malicious users controlled by the attacks can always simulate the original distribution.
- Hard sampling is implemented with the recommendation scores of items in each round. Actually, as we will show in Section 4 this approach may work well in the later stage of the training but fail in the early rounds when model parameters are still not poorly optimised.
- Previous works underestimate the power of the adversary in the sense that he/she can coordinate the communication between malicious users. When malicious users share information and collude with each other, the combined efforts can cause more detrimental damage to the FR system compared to independent attacks.

4 OUR FRecAttack^2 ATTACK

We will describe in detail how to explore more comprehensively the interplay between general user profiles and item embeddings to implement the two building blocks of an untargeted attack, respectively. We start by formally defining our threat model in terms of the adversary's knowledge.

4.1 Adversary knowledge

In general, we only assume the adversary knows the information that is accessible to malicious users. And they do not have any prior knowledge about users and items. Specifically, the adversary's knowledge consists of:

- the recommendation function, i.e., ψ;
- the knowledge of each malicious user $\widehat{u} \in \widehat{\mathcal{U}}$, i.e.,
  - values of the parameters maintained and distributed by the aggregator, i.e., VP^{ℓ}_{agg} for each $\ell \le T$.
  - true values of the parameters calculated and considered sensitive by the malicious user, i.e., $VP^{\widehat{u},\ell}_{user}$ and $\nabla\mathcal{P}^{\widehat{u},\ell}_{user}$ for each $\ell \le T$.

As mentioned previously, we consider the power of adversary coordinating the communication between malicious users and enabling collusion between them. We only explore the preliminary form of collusion that malicious users share information with others.

Table 1: Classification of attack scenarios against FR.

Attack type	independent attack	Type of FR	
		secret-hiding	DP-based
IndSH	✓	✓	
ColSH	✗	✓	
IndDP	✓		✓
ColDP	✗		✓

4.2 Virtual user embedding sampling

Recall that the purpose of user sampling is to provide a set of user profile embeddings that can approximately simulate the distribution

of benign users. We identify four attack scenarios considering the privacy protection mechanisms of FR (secret-hiding or DP-based) and the adversary's power to promote collusion among malicious users (colluding or independent attack). In Table 1, we list the names that will be used in this paper. We will describe the methods of virtual user sampling in detail for each attack scenario. As user sampling is run on each selected malicious user, we take a user $\widehat{u} \in \widehat{\mathcal{U}}$ as an example.

4.2.1 `IndSH`. In this type of attack, a malicious user conducts the user sampling independently without communicating with other malicious users. With the limited amount of local knowledge, each selected malicious user can only sample users in the neighbourhood of its own representation. Alg. 2 summarises the main steps. At any round ℓ, the malicious user $\widehat{u}$ first calculates his/her real representation without any attack by optimising the original loss function with the received parameters, i.e., the item representations $\mathbf{Y}_\ell$ and the prediction function parameters $\mathbf{\Theta}_\psi^\ell$ as well as a learning rate λ. This learning rate may not be the same as the one used by the aggregator (line 2-3). Then the user $\widehat{u}$ draws n samples from a normal distribution with its representation $\widehat{\mathbf{x}}^{\ell+1}$ as the mean and a hyperparameter σ^2 as the variance (line 5). The parameter n is pre-defined and all malicious users share the same value. Note that we use n to define the absolute number of sampled users instead of a percentage because in practice, the adversary does not know the total number of users.

Algorithm 2: User sampling for `IndSH`.

1 **Function** `SampleUserInd-SH`$(\mathbf{Y}^\ell, \mathbf{\Theta}_\psi^\ell, \lambda)$:
2 $\quad \nabla \widehat{\mathbf{x}}_{\widehat{u}}^{\ell+1} = \arg\min_{\nabla \mathbf{x}, \nabla \mathbf{Y}, \nabla \mathbf{\Theta}_\psi} Loss(\widehat{\mathbf{x}}_{\widehat{u}}^\ell, \mathbf{Y}^\ell, \mathcal{D}_{\widehat{u}}, \mathbf{\Theta}_\psi^\ell)$
3 $\quad \widehat{\mathbf{x}}_{\widehat{u}}^{\ell+1} = \widehat{\mathbf{x}}_{\widehat{u}}^\ell + \lambda \cdot \nabla \widehat{\mathbf{x}}_{\widehat{u}}^{\ell+1}$
4 $\quad$ **for** $i = 1 \dots n$ **do**
5 $\quad\quad$ `sample` $\mathbf{x}_i \sim \mathcal{N}(\widehat{\mathbf{x}}_{\widehat{u}}^{\ell+1}, \sigma^2)$
6 $\quad\quad \mathbf{X}_{\widehat{\mathcal{U}}} = \mathbf{X}_{\widehat{\mathcal{U}}} \cup \{\mathbf{x}_i\}$
7 $\quad$ **end**
8 $\quad$ **return** $\mathbf{X}_{\widehat{\mathcal{U}}}$

Discussion. This user sampling method works best when benign users can be clustered into groups of the same size and density, and each malicious user can occupy the centre of a group. Obviously, this is not realistic with such limited prior knowledge. However, when malicious users' profile embeddings can form a similar distribution to the true distribution of benign users, this method can give a better approximation compared to the ones used in previous research which solely use the malicious user's profile embedding.

4.2.2 `ColSH`. We describe the user sampling function to attack secret-hiding FR models with colluding malicious users. Compared to `IndSH`, in `ColSH`, the aggregator coordinates data sharing among malicious users. In secret-hiding FR systems, the only information that can be shared is the original user embeddings of malicious users. Despite the limited number of malicious users, these embeddings can still provide additional information to improve the inference of the user embedding distributions of benign users. In general, the idea is to cluster malicious users' original embeddings into groups. Then each group can be considered to represent a cluster of benign users. With each cluster centre calculated, a set of virtual user embeddings are sampled from a normal distribution with the mean and co-variance computed based on the malicious users' embeddings in the cluster. Instead of assigning a fixed number of sampled user embeddings, the number of samples drawn in each cluster should be proportional to its size. Alg. 3 depicts the corresponding steps to attack in `ColSH`. After finishing the update of the original user embedding (line 2–3), the malicious user $\widehat{u}$ sends it to the attacker who will gather and distribute all other malicious users' embeddings (line 5). Upon the receipt of the distributed malicious users' embeddings, the malicious user clusters them into groups with a clustering algorithm (line 7). In our implementation, we make use of the classic algorithm k-means [20]. However, instead of giving a single pre-defined number of clusters, we adopt a flexible implementation by automatically selecting the best number of clusters in a given set of candidate values. For each cluster, the user calculates the mean and co-variance (line 9–10) as well as the number of samples (line 11) which is proportional to the cluster size. Note n is the total number of samples as given in `IndSH`. Finally, n_C user embeddings are sampled.

Algorithm 3: User sampling for `ColSH`.

1 **Function** `SampleUserCol-SH`$(\mathbf{Y}^\ell, \mathbf{\Theta}_\psi^\ell, \lambda)$:
2 $\quad \nabla \widehat{\mathbf{x}}_{\widehat{u}}^{\ell+1} = \arg\min_{\nabla \mathbf{x}, \nabla \mathbf{Y}, \nabla \mathbf{\Theta}_\psi} Loss(\widehat{\mathbf{x}}_{\widehat{u}}^\ell, \mathbf{Y}^\ell, \mathcal{D}_{\widehat{u}}, \mathbf{\Theta}_\psi^\ell)$
3 $\quad \widehat{\mathbf{x}}_{\widehat{u}}^{\ell+1} = \widehat{\mathbf{x}}_{\widehat{u}}^\ell + \lambda \cdot \nabla \widehat{\mathbf{x}}_{\widehat{u}}^{\ell+1}$
4 $\quad$ `SendToAttacker`$(\widehat{\mathbf{x}}_{\widehat{u}}^{\ell+1})$
5 $\quad$ `wait until receiving from attacker` $\widehat{\mathbf{X}}_{\widehat{\mathcal{U}}} = \{\widehat{\mathbf{x}}_{\widehat{u}'} | \widehat{u}' \in \widehat{\mathcal{U}} / \{\widehat{u}\}\}$
6 $\quad \widehat{\mathbf{X}}_{\widehat{\mathcal{U}}} = \{\widehat{\mathbf{x}}_{\widehat{u}}^{\ell+1}\} \cup \widehat{\mathbf{X}}_{\widehat{\mathcal{U}}}$
7 $\quad C = \texttt{cluster}(\widehat{\mathbf{X}}_{\widehat{\mathcal{U}}})$
8 $\quad$ **for** *cluster* $C \in C$ **do**
9 $\quad\quad \widehat{\mathbf{x}}_C = \frac{1}{|C|} \sum_{\widehat{\mathbf{x}}_{\widehat{u}'} \in C} \widehat{\mathbf{x}}_{\widehat{u}'}$
10 $\quad\quad \sigma_C = \texttt{CaculateCoVar}(\mathbf{X}_C)$
11 $\quad\quad n_C = n \cdot \frac{|C|}{\sum_{C' \in C} |C'|}$
12 $\quad\quad$ **for** $i = 1, \dots, n_C$ **do**
13 $\quad\quad\quad$ `sample` $\mathbf{x}_i \sim \mathcal{N}(\widehat{\mathbf{x}}_C, {\sigma_C}^2)$
14 $\quad\quad\quad \mathbf{X}_{\widehat{\mathcal{U}}} = \mathbf{X}_{\widehat{\mathcal{U}}} \cup \{\mathbf{x}_i\}$
15 $\quad\quad$ **end**
16 $\quad$ **end**
17 $\quad$ **return** $\mathbf{X}_{\widehat{\mathcal{U}}}$

4.2.3 `IndDP`. Different from secret-hiding FR systems, in a DP-based FR system, each user has access to the noisy version of all three types of parameters including user embeddings in each round ℓ, i.e., $\widetilde{\mathbf{X}}^\ell$. Compared to `IndSH` where a malicious user only samples local user embeddings centred around his/her original embedding, in `IndDP`, the user can draw user samples with the *noisy* global distribution of benign users. Although the original embedding of the malicious user could provide certain useful information, just one true user profile embedding fails to make a sufficient difference. We adopt the efficient solution by directly clustering the noisy user embeddings in $\mathbf{X}^\ell$ and drawing samples in a normal distribution calculated according to each cluster. We demonstrate this method in Alg. 4. We cluster the noisy embeddings of users with the same clustering algorithm as that used in `ColSH`. Then for each cluster,

we calculate the centre and the covariance vector, respectively, as well as the number of virtual users to sample (line 4–5). In the end, we draw n_C samples for the normal distribution tailored to each cluster C as the virtual user embeddings. Note that the number of samples of a cluster is proportional to the size of the cluster.

Algorithm 4: User sampling for IndDP.

1 **Function** SampleUserInd-DP($\widetilde{\mathbf{X}}^{\ell}$):
2 C = cluster($\widetilde{\mathbf{X}}^{\ell}$)
3 **for** *cluster* $C \in C$ **do**
4 $\widehat{\mathbf{x}}_C = \frac{1}{|C|} \sum_{\mathbf{x}_u \in C} \widetilde{\mathbf{x}}_u$
5 σ_C = CaculateCoVar($\widetilde{\mathbf{X}}_C$)
6 $n_C = n \cdot \frac{|C|}{\sum_{C' \in C} |C'|}$
7 **for** $i = 1, \ldots, n_C$ **do**
8 sample $\mathbf{x}_i \sim \mathcal{N}(\widehat{\mathbf{x}}_C, {\sigma_C}^2)$
9 $\mathbf{X}_{\ddot{\mathcal{U}}} = \mathbf{X}_{\ddot{\mathcal{U}}} \cup \{\mathbf{x}_i\}$
10 **end**
11 **end**
12 **return** $\mathbf{X}_{\ddot{\mathcal{U}}}$

4.2.4 ColDP. In this attack scenario, malicious users' true profile embeddings cannot be ignored as IndDP considering their relatively larger volume. The problem is reduced to how to use the two sources of user profile embeddings to reduce the noise added by the differential privacy methods. The clusters calculated only with the perturbed user embeddings in IndDP are also noisy. However, they provide us with a relatively reliable estimation of the groups of users. If we can identify the group every malicious user belongs to, then we can use the true embeddings of malicious users in the same group to re-calculate the mean and covariance parameters of a normal distribution to approximate the distribution of user profiles in that group. The general idea is to assign any malicious user to a group according to the closest noisy cluster centre to his/her true profile embedding. Our method is presented in Alg. 5. After receiving other malicious users' true embeddings from the attacker, the malicious user $\widehat{u}$ clusters the benign users with their noisy profile embeddings (line 7) and calculates the centre of each cluster (line 9). Then each malicious user is assigned to the cluster whose centre has the minimum distance from him/her (line 11-13). For each new group of malicious users, the mean and covariance of a normal distribution are calculated according to their true embeddings (line 15-18). Then virtual user profile embeddings are sampled from the distribution. The number of samples is proportional to the size of the group of malicious users.

4.3 Interaction sampling

With user sampling presented above, we have a set of sampled users, i.e., $\ddot{\mathcal{U}}$, and their profile embeddings $\mathbf{X}_{\ddot{\mathcal{U}}}$. We proceed to assign each sampled virtual user, e.g., $\ddot{u} \in \ddot{\mathcal{U}}$, a set of past user-item interactions, i.e., $\ddot{\mathcal{D}}_{\ddot{u}}$. As discussed previously, the general idea is to find the hardest negative and positive interactions of every virtual user. Existing works rely on absolute recommendation ratings. Specifically, the ones with the top recommendation ratings are used as the hardest negative interactions and those with bottom scores as the hardest positive interactions. This approach may work in the later rounds during training when the recommendation function is close to optimal. However, it may fail to remain effective at the early stage of the training. We start by using two widely used real-life datasets described in our Section 4.3.1 to validate our hypothesis and then present our solution according to the validation.

Algorithm 5: User sampling for ColDP.

1 **Function** SampleUserCol-DP($\widetilde{\mathbf{X}}^{\ell}$, $\mathbf{Y}^{\ell}$, $\mathbf{\Theta}^{\ell}_{\psi}$, λ):
2 $\nabla \widehat{\mathbf{x}}^{\ell+1}_{\widehat{u}} = \arg\min_{\nabla \mathbf{x}, \nabla \mathbf{Y}, \nabla \mathbf{\Theta}_{\psi}} Loss(\widehat{\mathbf{x}}^{\ell}_{\widehat{u}}, \mathbf{Y}^{\ell}, \mathcal{D}_{\widehat{u}}, \mathbf{\Theta}^{\ell}_{\psi})$
3 $\widehat{\mathbf{x}}^{\ell+1}_{\widehat{u}} = \widehat{\mathbf{x}}^{\ell}_{\widehat{u}} + \lambda \cdot \nabla \widehat{\mathbf{x}}^{\ell+1}_{\widehat{u}}$
4 SendToAttacker($\widehat{\mathbf{x}}^{\ell+1}_{\widehat{u}}$)
5 wait until receiving from attacker $\widehat{\mathbf{X}}_{\widehat{\mathcal{U}}} = \{\widehat{\mathbf{x}}_{\widehat{u}'} | \widehat{u}' \in \widehat{\mathcal{U}} / \{\widehat{u}\}\}$
6 $\widehat{\mathbf{X}}_{\widehat{\mathcal{U}}} = \{\widehat{\mathbf{x}}^{\ell+1}_{\widehat{u}}\} \cup \widehat{\mathbf{X}}_{\widehat{\mathcal{U}}}$
7 C = cluster($\widetilde{\mathbf{X}}^{\ell}$)
8 **for** *cluster* $C \in C$ **do**
9 $\mathbf{x}_C = \frac{1}{|C|} \sum_{\widetilde{\mathbf{x}}_u \in C} \widetilde{\mathbf{x}}_u$
10 **end**
11 **for** *malicious user* $\widehat{u} \in \widehat{\mathcal{U}}$ **do**
12 $c_{\widehat{u}} = \arg\min_{C \in C}$ distance($\widehat{\mathbf{x}}_{\widehat{u}}, \mathbf{x}_C$)
13 **end**
14 **for** *cluster* $C \in C$ **do**
15 $\widehat{\mathcal{U}}_C = \{\widehat{u} \in \widehat{\mathcal{U}} | c_{\widehat{u}} = C\}$
16 $\widehat{\mathbf{X}}_{\widehat{\mathcal{U}}_C} = \{\widehat{\mathbf{x}}_{\widehat{u}'} | \widehat{u}' \in \widehat{\mathcal{U}}_C\}$
17 $\widehat{\mathbf{x}}_C = \frac{1}{|\widehat{\mathcal{U}}_C|} \sum_{\widehat{\mathbf{x}}_{\widehat{u}'} \in \widehat{\mathbf{X}}_{\widehat{\mathcal{U}}_C}} \widehat{\mathbf{x}}_{\widehat{u}'}$
18 σ_C = CaculateCoVar($\widehat{\mathbf{X}}_{\widehat{\mathcal{U}}_C}$)
19 $n_C = n \cdot \frac{|\widehat{\mathcal{U}}_C|}{\sum_{C' \in C} |\widehat{\mathcal{U}}_{C'}|}$
20 **for** $i = 1, \ldots, n_C$ **do**
21 sample $\mathbf{x}_i \sim \mathcal{N}(\widehat{\mathbf{x}}_C, {\sigma_C}^2)$
22 $\mathbf{X}_{\ddot{\mathcal{U}}} = \mathbf{X}_{\ddot{\mathcal{U}}} \cup \{\mathbf{x}_i\}$
23 **end**
24 **end**
25 **return** $\mathbf{X}_{\ddot{\mathcal{U}}}$

4.3.1 Data validation. We train a typical secret-hiding model FedNCF [12] on the Steam dataset and a DP-based FR model FedSoG [19] on the Filmtrust dataset. Specifically, we quantify the prediction accuracy of hard samples by measuring the difference between the sets of true hardest samples (both positive and negative) after a normal training and the sets predicted during the training in each round. Let $\mathcal{R}^{+}_{u}$ be the top K items with the largest recommendation ratings and $\mathcal{R}^{-}_{u}$ be the K items with the smallest ratings when FR is normally trained without any malicious intervention. We use $\mathcal{R}^{+,\ell}_{u}$ and $\mathcal{R}^{-,\ell}_{u}$ to denote the corresponding sets of samples determined by the recommendation score in round ℓ. Then the accuracy for positive interactions of user u at the round ℓ is calculated as the recall rate of true hard samples, i.e., $\frac{|\mathcal{R}^{+}_{u} \cap \mathcal{R}^{+,\ell}_{u}|}{|\mathcal{R}^{+}_{u}|}$. Similarly, we can have the accuracy for negative samples. We calculate the average accuracy of all users as our final measurement for prediction accuracy denoted by Acc^{+}_{ℓ} for hardest positive samples and Acc^{-}_{ℓ} for hardest negative samples. In Figure 2, we show the average accuracy changes when the hardest samples are predicted only according to recommendation ratings during training with the

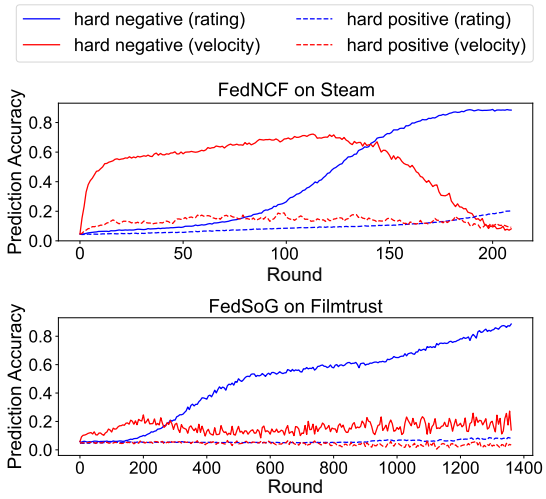

Figure 2: Hard item sampling with recommendation ratings and their change velocities.

blue colour. We can see that for both the algorithms the prediction accuracy for hard negative samples can gradually reach over 0.8 in the final stage of training, but is rather low in the early stage although the improving speed may differ between algorithms and datasets. The accuracy remains below 0.2 even until half of the training is finished. The prediction accuracy for negative samples remains at a comparatively low level due to the large number of negative samples with close recommendation ratings. However, the same pattern as that of positive samples also presents.

From the observation, we can conclude that the hard sampling method used in recent works [32] *cannot* effectively identify the correct hardest positive and negative samples in the early stage of training and this will potentially damage the final attack performance. In this paper, we propose a new method to sample the hard samples. Intuitively, instead of the absolute recommendation ratings, we leverage the velocity of their changes between rounds.

We can interpret the iterative optimisation as a process to increase the difference of the recommendation ratings of real hard negative samples from the hard negative ones. As a result, the recommendation score of hard negative samples will *increase more quickly* than hard positive samples. We make use of this idea and propose a method based on the changing velocity of recommendation ratings in recent rounds. Note that we use 'velocity' instead of 'speed' because compared to 'speed', velocity measures a change in two aspects: direction and magnitude. Regarding recommendation score changes, the *direction* of the velocity is positive when they increase and negative when they decrease. The *magnitude* measures the size of the change. We also illustrate the performance of the hard sampling based on recommendation score change velocity through the same datasets and FR models. The results are also shown in Figure 2 with the red colour for a clear comparison between the two measurements. We can see that different from the blue curves, change velocity can lead to a significantly higher level of prediction accuracy in the early training stage. However, the prediction performance starts to plunge in the late stage. This actually confirms our interpretation of model optimisation. When the training is close to converging, recommendation ratings are rather stable and too indistinguishable to identify hard positive and negative samples. Based on the above discussion, we propose combining the two measurements: absolute recommendation ratings and their change velocities to enforce a more accurate sampling.

4.3.2 Velocity-based interaction sampling. For each sampled user $\ddot{u} \in \ddot{\mathcal{U}}$, the goal of the malicious user is to infer the most likely positive and negative interactions and then swap the labels to manipulate SGD updates. To achieve this goal, malicious users need to predict the final recommendation rankings. Considering the better performance of predictions based on change velocities in the early training stage, our idea is to rely on change velocities from the beginning of the training and make a switch to predictions according to recommendation ratings at a *'right moment'*. Next, we will describe in detail how to make predictions with change velocities and how to dynamically determine the '*right*' switch moment.

V-based **interaction ranking.** We take the round ℓ ($\ell > 0$) as an example to illustrate the calculation. We introduce a sliding window to dynamically define the previous round to compare when velocity is calculated. Let w be the window size. Recall $\psi^{\ell}(\mathbf{x}, \mathbf{y})$ denotes the recommendation function instantiated with the parameter $\mathbf{\Theta}^{\ell}_{\psi}$ distributed by the aggregator to users before the round ℓ starts. We calculate the recommendation score change velocity of item η for the sampled user $\ddot{u}$ as follows:

$$\Delta^{\ell}_{\ddot{u},\eta} = \begin{cases} \frac{\psi^{\ell}(\mathbf{x}_{\ddot{u}}, \mathbf{y}^{\ell}_{\eta}) - \psi^{\ell-w}(\mathbf{x}_{\ddot{u}}, \mathbf{y}^{\ell-w}_{\eta})}{w}, & \text{if } \ell > w; \\ \frac{\psi^{\ell}(\mathbf{x}_{\ddot{u}}, \mathbf{y}^{\ell}_{\eta}) - \psi^{0}(\mathbf{x}_{\ddot{u}}, \mathbf{y}^{0}_{\eta})}{\ell}, & \text{if } 0 < \ell \leq w. \end{cases}$$

Note that $\mathbf{y}^{0}_{\eta}$ is the initial representation vector of item η. As user $\ddot{u}$ is virtual and does not exist in practice, we have no access to its profile representation vector in any previous round. As a result, we use the sampled profile embedding $\mathbf{x}_{\ddot{u}}$ as an approximation in the definition. Ideally, with the above measurement, we can obtain a personal ranking of items in $\mathcal{I}$ for each user. However, this personalisation actually negatively impacts the effectiveness of the untargeted attack. This is because it leads to a complex attack loss function with too many parameters to optimise. The employed optimisation function subsequently gets its performance degraded, especially to deal with possible conflicting preferences from users with similar profiles. As a result, we calculate a uniform ranking shared by all sampled virtual users instead to further simplify the attack loss function. The ranking is based on the sum of the recommendation ratings of all virtual users for each item. As a result, instead of $\Delta^{\ell}_{\ddot{u},v}$, we use $\Delta^{\ell}_{\eta} = \sum_{\ddot{u} \in \ddot{\mathcal{U}}} \Delta^{\ell}_{\ddot{u},\eta}$ to sort the items descendingly. With this ranking, we can construct two sets. One is the set of the hardest negative samples $\mathcal{R}^{-}_{v}$ while the other is the set of hardest positive samples $\mathcal{R}^{+}_{v}$. The sizes of both sets are $\gamma|\mathcal{I}|$ where γ is a hyper-parameter defining the percentage of items to be selected as hard positive or negative samples. Note we can also set different values for negative samples and positive samples. With these two sets, we can obtain the manipulated past item interactions for all virtual users denoted by $\ddot{\mathcal{D}}_{\Delta}$ as

$$\ddot{\mathcal{D}}^{\ell}_{\Delta} = \{(\ddot{u}, \eta, 0) | \eta \in \mathcal{R}^{-}_{v} \wedge \ddot{u} \in \ddot{\mathcal{U}}\} \cup \{(\ddot{u}, \eta, 1) | \eta \in \mathcal{R}^{+}_{v} \wedge \ddot{u} \in \ddot{\mathcal{U}}\}.$$

Algorithm 6: Virtual Interaction sampling.

Function SampleItemInteraction($\ddot{\mathcal{U}}, \mathbf{X}^\ell, \mathbf{Y}^\ell, \Theta^\ell_\psi$):
  for *each item* $\eta \in \mathcal{I}$ **do**
    $\Delta^\ell_\eta = 0, \Psi^\ell_\eta = 0$
    for *each user* $\ddot{u} \in \ddot{\mathcal{U}}$ **do**
      calculate $\Delta^\ell_{\ddot{u},\eta}$
      $\Delta^\ell_\eta = \Delta^\ell_\eta + \Delta^\ell_{\ddot{u},\eta}$
      $\Psi^\ell_\eta = \Psi^\ell_\eta + \psi^\ell(\mathbf{x}_{\ddot{u}}, \mathbf{y}_\eta)$
    end
  end
  $\mathcal{R}^+_\Delta$ = BottomN($\Delta^\ell_\eta : \eta \in \mathcal{I}$)
  $\mathcal{R}^-_\Delta$ = TopN($\Delta^\ell_\eta : \eta \in \mathcal{I}$)
  $\ddot{\mathcal{D}}^\ell_\Delta$ = ConstructInteractionSet($\mathcal{R}^+_\Delta, \mathcal{R}^-_\Delta$)
  $\mathcal{R}^+_\Psi$ = BottomN($\psi^\ell(\mathbf{x}_{\ddot{u}}, \mathbf{y}_\eta) : \eta \in \mathcal{I}$)
  $\mathcal{R}^-_\Psi$ = TopN($\psi^\ell(\mathbf{x}_{\ddot{u}}, \mathbf{y}_\eta) : \eta \in \mathcal{I}$)
  $\ddot{\mathcal{D}}^\ell_\Psi$ = ConstructInteractionSet($\mathcal{R}^+_\Psi, \mathcal{R}^-_\Psi$)
  $\ddot{\mathcal{D}}^\ell = \ddot{\mathcal{D}}^\ell_\Delta, s_0 = -\infty, m = 0, i = 1$
  if $\ell > M$ **then**
    $i = \ell - M + 1$
  end
  while $i \leq \ell$ **do**
    $s_i = |\ddot{\mathcal{D}}^i_\Delta \cap \ddot{\mathcal{D}}^{i-1}_\Delta|$
    if $s_i - s_{i-1} < 0$ **then**
      $m = m + 1$
    end
    $i = i + 1$
  end
  if $m < \frac{M}{2}$ **then**
    $\ddot{\mathcal{D}}^\ell = \ddot{\mathcal{D}}_\Psi$
  end
  return $\ddot{\mathcal{D}}^\ell$

Similarly, if we replace recommendation score change velocities with recommendation ratings calculated by the recommendation score function ψ^ℓ, we have another set of past item interactions denoted by $\ddot{\mathcal{D}}^\ell_\Psi$.

Switch between $\ddot{\mathcal{D}}^\ell_\Delta$ and $\ddot{\mathcal{D}}^\ell_\Psi$. As we discussed previously, the V-based method gives a prediction of hard samples with an accuracy souring fast from the beginning of the training but falling off in the late stage while the accuracy generated by recommendation ratings keeps increasing and reaches a high level in the late stage. As a result, if we can identify the point where the V-based method's performance falls, then we can have the right moment for the switch to absolute recommendation ratings. We use a heuristic approach to identifying this falling point. When the prediction accuracy increases, more sampled hard items (both positive and negative) will be shared between consecutive rounds. In other words, the size of $\ddot{\mathcal{D}}^{\ell-1}_\Delta \cap \ddot{\mathcal{D}}^\ell_\Delta$ increases when ℓ grows. However, as the prediction accuracy fluctuates during training, this consecutive increase cannot occur in each round. As a consequence, we consider a window with the last M rounds, i.e., from round $\ell - M$ to ℓ. If the shared items between two consecutive rounds increase in less than half of M previous rounds, then we consider the V-based method's performance starts falling and make the switch from $\ddot{\mathcal{D}}_\Delta$ to $\ddot{\mathcal{D}}_\Psi$. In Alg. 6, we show the corresponding pseudocodes.

5 EXPERIMENTAL EVALUATION

5.1 Experiment settings

Datasets. We evaluate our attacks with four real-life datasets: Movielens-1M (ML-1M), Steam-200K(Steam), Lastfm and Filmtrust, representing different domains respectively (e.g., movies, games, music). These datasets have been widely exploited in empirical analysis of attacks on FR systems [8, 34, 39] which only contain users' item interactions. As secret-hiding and DP-based FR models require different data input, we use different datasets to evaluate our attack on these two types, i.e., Steam and ML-1M for secret-hiding models and Lastfm and Filmtrust for DP-based models. The statistics of the datasets are shown in Table 2. Note that density is measured as the proportion of observed user-item and user-user interactions among all possible combinations.

Table 2: Statistics of datasets.

Dataset	ML-1M	Steam	Lastfm	Filmtrust
#Users	6,040	3,753	1,892	874
#Items	3,706	5,134	17,632	1,957
#Ratings	1,000,209	114,713	92,834	18,662
#Social Connection	-	-	5,676	1,853
Rating Density	4.47%	0.59%	0.28%	1.09%
Connection Density	-	-	0.16%	0.24%

Victim FR models. We evaluate the effectiveness of our attack on two secret-hiding FR models: FedNCF [24] and FedMLP [24], and two DP-based models: FedGNN [31] and FedSoG [19]. FedNCF and FedMLP belong to collaborative filtering systems which are widely employed in real-world recommender systems. In particular, FedNCF is based on NCF [12] which is the most typical centralised deep learning approach for recommendation. FedGNN and FedSoG are the two most recent works that combine graph neural networks with FR systems while providing a DP guarantee.

Attack benchmarks. We compare our attack to the following state-of-the-art untargeted attacks: (1) **SignFlip** [16], flipping the direction of normal gradients of malicious clients; (2) **FedAttack** [32], selecting the most similar items to user embeddings as the hardest negative samples and the least similar ones as the hardest positive samples; (3) **Gaussian** [10], uploading SGD updates drawn from an estimated Gaussian distribution with the mean and standard deviation of the normal gradients; (4) **LIE** [2], adding small amounts of noise to the average of normal gradients; and (5) **ClusterAttack** [37], clustering item embeddings and minimising inter-distance between item embeddings in the same cluster.

Experimental parameter setting We employ the standard horizontal federated learning framework for training [35], where the aggregator randomly selects 10% of the clients to participate in each round of training. Each user is treated as a client in the FR system. The user profile embedding and item embedding dimensions are set to 16. Note that the negative interactions are down-sampled using a positive-to-negative ratio of 1 : 1 due to the large number of negative interactions. We conduct our experiments on a Ubuntu server with 4 NVIDIA GeForce RTX 4090 GPUs, a 64-bit 20-core Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz and 125GBs of RAM. For the attack scenarios without colluding malicious users,

Table 3: Degrading performance with no defences.

Dataset	Method	FedNCF		FedMLP	
		HR@10	NDCG@10	HR@10	NDCG@10
ML-1M	No Attack	0.0850 (-)	0.0434 (-)	0.0838 (-)	0.0421 (-)
	SignFlip	0.0841 (1.06%)	0.0429 (1.15%)	0.0827 (1.31%)	0.0413 (1.90%)
	LabelFlip	0.0827 (2.71%)	0.0414 (4.61%)	0.0840 (-0.24%)	0.0423 (-0.48%)
	FedAttack	0.0824 (3.06%)	0.0417 (3.92%)	0.0803 (4.18%)	0.0400 (4.99%)
	Gaussian	0.0836 (1.65%)	0.0421 (2.99%)	0.0827 (1.31%)	0.0422 (-0.24%)
	LIE	0.0833 (2.00%)	0.0423 (2.53%)	0.0820 (2.15%)	0.0414 (1.66%)
	ClusterAttack	0.0814 (4.24%)	0.0434 (0.00%)	0.0818 (2.39%)	0.0408 (3.09%)
	FRecAttack^2-IndSH	0.0780 (8.24%)	0.0383 (11.75%)	0.0793 (5.37 %)	0.0399 (5.23%)
	FRecAttack^2-ColSH	**0.0753 (11.41%)**	**0.0381 (12.21%)**	**0.0746 (10.98%)**	**0.0377 (10.45%)**
Steam	No Attack	0.1903 (-)	0.1110 (-)	0.1783 (-)	0.1002 (-)
	SignFlip	0.1805 (5.15%)	0.1092 (1.62%)	0.1733 (2.80%)	0.0932 (6.99%)
	LabelFlip	0.1808 (4.99%)	0.1017 (8.38%)	0.1736 (2.64%)	0.0940 (6.19%)
	FedAttack	0.1853 (2.63%)	0.0999 (10.00%)	0.1738 (2.52%)	0.0903 (9.88%)
	Gaussian	0.1877 (1.37%)	0.1071 (3.51%)	0.1759 (1.35%)	0.0955 (4.69%)
	LIE	0.1881 (1.16%)	0.1090 (1.80%)	0.1761 (1.23%)	0.0944 (5.79%)
	ClusterAttack	0.1804 (5.20%)	0.1018 (8.29%)	0.1735 (2.69%)	0.0931 (7.09%)
	FRecAttack^2-IndSH	0.1736 (8.78%)	0.0981 (11.62%)	0.1728 (3.08%)	0.0891 (11.08%)
	FRecAttack^2-ColSH	**0.1660 (12.77%)**	**0.0931 (16.13%)**	**0.1529 (14.25%)**	**0.0805 (19.66%)**
Dataset	**Attack**	**FedSoG**		**FedGNN**	
		HR@10	NDCG@10	HR@10	NDCG@10
Lastfm	No Attack	0.0288 (-)	0.0150 (-)	0.0235(-)	0.0119 (-)
	SignFlip	0.0242 (15.97%)	0.0119 (20.66%)	\	\
	LabelFlip	0.0256 (11.11%)	0.0122 (18.67%)	0.0218 (7.23%)	0.0111 (6.72%)
	FedAttack	0.0233 (19.10%)	0.0117 (22.00%)	0.0207 (11.91%)	0.0102 (14.29%)
	Gaussian	0.0262 (9.03%)	0.0142 (5.33%)	0.0231 (1.70%)	0.0115 (3.36%)
	LIE	0.0254 (11.81%)	0.0139 (7.33%)	0.0223 (5.11%)	0.0109 (8.40%)
	FRecAttack^2-IndDP	0.0221 (23.26%)	0.0101 (32.67%)	0.0185 (21.27%)	0.0098 (17.65%)
	FRecAttack^2-ColDP	**0.0212 (26.39%)**	**0.0094 (37.33%)**	**0.0176 (25.11%)**	**0.0092(22.69%)**
Filmtrust	No Attack	0.5022 (-)	0.3003 (-)	0.5276 (-)	0.3210 (-)
	SignFlip	0.4087 (18.62%)	0.1849 (38.43%)	\	\
	LabelFlip	0.4984 (0.76%)	0.2786 (7.23%)	0.5033 (4.61%)	0.2886 (10.09%)
	FedAttack	0.4066 (19.11%)	0.1843 (38.63%)	0.4376 (17.05%)	0.2764 (13.89%)
	Gaussian	0.4835 (3.74%)	0.2550 (15.08%)	0.5083 (3.66%)	0.2914 (9.22%)
	LIE	0.4797 (4.50%)	0.2380 (20.75%)	0.4870 (7.70%)	0.2731 (14.92%)
	FRecAttack^2-IndDP	0.3744 (25.45%)	0.1644 (44.59%)	0.3895 (26.18%)	0.2496 (22.24%)
	FRecAttack^2-ColDP	**0.3638 (27.56%)**	**0.1553 (48.29%)**	**0.3890 (26.27%)**	**0.2378 (25.92%)**

i.e., FRecAttack^2-IndSH and FRecAttack^2-IndDP, the covariance of the normal distribution σ^2 used in virtual user sampling is set to ensure the square of its every dimension equivalent to 0.1. We select the optimal learning rates for the selected FR models by parameter tuning and our attack adopts the same learning rates.

Evaluation measurements. We use the Hit Ratio (HR) and the Normalized Discounted Cumulative Gain (NDCG) over the top K ranked items to evaluate the effectiveness of untargeted attacks in degrading recommendation performance. For each user, HR@Kmeasures the proportion of recommended items that also appear in the user's true top K recommendations. It evaluates the ability of the FR system to make correct recommendations for benign users. Compared to HR@K, NDCG takes into account the rankings of recommended items, i.e., their positions in the final recommendation list. It additionally measures the quality of ranking of a recommender system. Note that all values of both measurements are calculated for benign users. We select the value of 10 for K. To highlight the effectiveness of untargeted attacks, we also calculate the ratio of performance change after attacks, called *degrading ratio*. Let δ and δatk be the performance evaluated based on either HR@Kor NDCG without and with untargeted attacks, respectively. Then the ratio is calculated as $\frac{\delta-\delta atk}{\delta}$. All the numbers reported in this Section are averaged over 5 times of training, considering the potential instability of deep learning-based FR systems.

5.2 Overall performance evaluation

We validate the effectiveness of our FRecAttack^2 and compare it to the benchmark attacks. To ensure a fair comparison, we set the hyperparameters of FRecAttack^2 according to basic values without selecting on purpose the values that favour our attack.

For instance, we set the window sizes for velocity calculation and measure the switch between recommendation ratings and their change velocities to 2. The number of virtual user embeddings n is 150 and the proportion of hard positive (negative) samples γ is set to 10%. For all the benchmark attacks, we adopt the parameter values that lead to the optimal performance and are claimed in the original works. We assume malicious users counted for 10% of the total users are randomly selected in each execution for all attacks.

The experimental results are presented in Table 3. Note that ClusterAttack is not shown when DP-based models are attacked because it cannot guarantee the convergence of training in most cases. This may be because it manipulates all the items' embeddings in every round and when DP noises are added, it disrupts the normal training process. The degrading ratio is also appended after the performance values evaluated by HR and NDCG. Note that we add the attack scenarios to FRecAttack^2 to indicate the version of our attack. The results of SignFlip on FedGNN are not available because the model training cannot converge due to the flipping signs.

We have three observations. First, our FRecAttack^2 significantly degrades the performance of all targeted FR models in all four attack scenarios and overwhelms all the baseline attacks. FRecAttack^2 degrades the recommendation performance by at least 5% when attacking secret-hiding models and the performance decrease even reaches over 20% for DP-based models. By contrast, among the selected attacks, the two most recent ones, i.e., ClusterAttack and FedAttack, perform the best but FRecAttack^2 can still cause up to twice as much as their decreases in almost all cases. This shows that exploring the interplay between user profiles and items effectively enforces the effectiveness of FRecAttack^2. Second, the magnitudes of the degrading performance vary according to the datasets and types of privacy protection mechanisms. When attacking FedNCF and FedMLP on two datasets with different sizes, i.e., ML-1M and Steam, FRecAttack^2 performs better on Steam which is relatively smaller when measured by HR@K. This means attacking large-scale recommender systems with more users and items is more challenging and more resources such as larger proportions of malicious users are required. In addition, according to our results, DP-based FR models are more susceptible to attacks compared to secret-hiding ones. The degrading ratios are about 25% for both of the selected DP-based models which is at least 2 times as much as the degrading performance on secret-hiding models. This vulnerability can be attributed to the access to all users' profile embeddings in DP-based systems. This means that despite the perturbing noise added, the mechanism of sharing user profiles still exposes valuable auxiliary information for the adversary to interfere with FR training. Third, colluding malicious users can better degrade recommendation performance compared to independent attacks. In all cases, our FRecAttack^2 with colluding malicious users beats the corresponding independent versions by at least 10% when attacking secret-hiding models. The relative increase of colluding attacks in attack performance is slightly smaller when attacking DP-based models. This is because access to benign users' noisy profile embeddings has already ensured good performance and the significance of malicious users' true profile embedding thus becomes less significant relatively. From the above analysis, we validate the effectiveness of leveraging the interplay between user profiles and items in downgrading FR systems' overall performance.

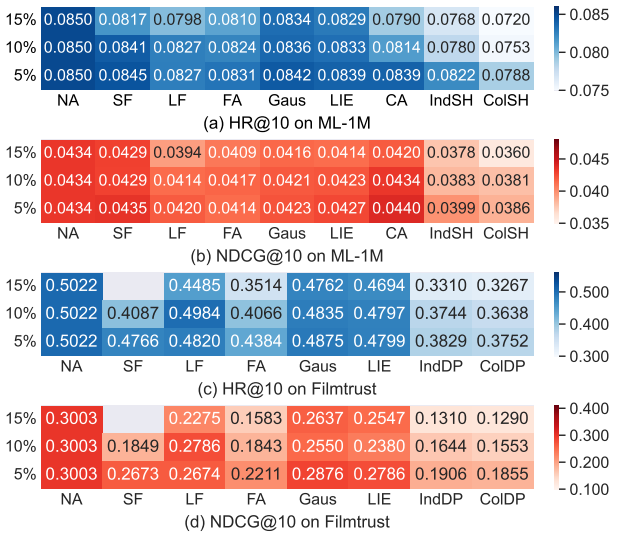

Figure 3: Attack performance with different proportions of malicious users (NA: No Attack, SF: SignFlip, LF: LabelFlip, FA: FedAttack, Gaus: Guassian, CA: ClusterAttack, IndDP: FRecAttack^2-IndDP, ColDP: FRecAttack^2-ColDP, IndSH: FRecAttack^2-IndSH, ColSH: FRecAttack^2-ColSH).

In addition, we also show the necessity to consider the power of the adversary in coordinating communication among malicious users for a comprehensive understanding of the potential threats to FR.

5.3 Impact of the proportion of malicious users

In the following experiments, we will use ML-1M and Filmtrust (the results with other datasets show similar trends and are omitted due to the space limit). Figure 3 shows the results of different poisoning attacks when different proportions of users are controlled by the adversary as malicious users (5%, 10%, and 15%). A lighter colour indicates a better degrading performance of an attack.

From the results, we observe that the performance of all attacks improves as the number of malicious users increases. For instance, in Figure 3(a), after compromising 5% malicious users in ML-1M, FRecAttack^2 with colluding malicious users can achieve a small hit ratio of 0.0788, while the hit ratio decreases by 4% when 10% users become malicious. In addition, we can see that the number of malicious users actually has more impact on the performance of FRecAttack^2 than others. This is because a larger number of malicious users will allow the adversary to obtain more prior information about the true user distribution. More importantly, our FRecAttack^2 can enforce the best performance in all cases. This indicates our attack unveils the true potential damage of untargeted attacks against FR even with a small number of malicious users.

5.4 Ablation study

In this subsection, we investigate the contribution of the two core components of our attack, i.e., user sampling and interaction sampling in attacking FR models. We also make use of the ML-1M and Filmtrust datasets and show the different variants of FRecAttack^2's

Table 4: Attack Effectiveness under our different attacks with 10% malicious users on ML-1M and Filmtrust.

FedNCF on ML-1M					
Attack	**HR@10**	**NDCG@10**	**HR@10**	**NDCG@10**	**Attack**
No Attack	0.0850 (-)	0.0434 (-)	0.0850 (-)	0.0434 (-)	No Attack
Gaus-Rating	0.0837 (1.53%)	0.0432 (0.46%)	0.0831 (2.24%)	0.0430 (0.92%)	Gaus-VRating
Single-Rating	0.0808 (4.94%)	0.0399 (8.06%)	0.0789 (7.18%)	0.0397 (8.53%)	Single-VRating
IndSH-Rating	0.0786 (7.53%)	0.0389 (10.37%)	0.0780 (8.24%)	0.0383 (11.75%)	IndSH-VRating
ColSH-Rating	**0.0755 (11.18%)**	**0.0383 (11.75%)**	**0.0753 (11.41%)**	**0.0381 (12.21%)**	ColSH-VRating
FedSoG on Filmtrust					
Attack	**HR@10**	**NDCG@10**	**HR@10**	**NDCG@10**	**Attack**
No Attack	0.5022 (-)	0.3003 (-)	0.5022 (-)	0.3003(-)	No Attack
Gaus-Rating	0.4973 (0.98%)	0.2954 (1.63%)	0.4902 (2.39%)	0.2811 (6.39%)	Gaus-VRating
Single-Rating	0.4942 (1.59%)	0.2667 (11.19%)	0.4687 (6.67%)	0.2260 (24.74%)	Single-VRating
IndDP-Rating	0.4269 (14.99%)	0.1973 (34.30%)	0.3744 (25.45%)	0.1644 (44.59%)	IndDP-VRating
ColDP-Rating	**0.3889 (22.56%)**	**0.1731 (42.35%)**	**0.3638 (27.56%)**	**0.1553 (48.29%)**	ColDP-VRating

performance when applied to attack FedNCF and FedSoG, respectively. For the user sampling module, in addition to our implementation in the four attack scenarios, we consider two additional simple baselines for comparison which only use the malicious user's true embedding or use embedding that follows the Gaussian distribution as the only virtual user. We denote the baselines by 'Single' and 'Gaus'. For the interaction sampling module, we consider the method relying on recommendation ratings, denoted by 'Rating'. Our method combines recommendation ratings and their change velocities. Therefore, we denote our item sampling method by 'VRating'. Each variant of our FRecAttack^2 attack is named by two parts connected by '–'. The left part indicates the implementation of user sampling while the right part indicates the selection of interaction sampling algorithms. The performance of various variants is presented in Table 4 which also includes the recommendation performance without any attacks for a clear illustration. The left half of Table 4 shows the recommendation performance when our attack is implemented with the interaction sampling solely based on recommendation ratings while the right half presents the corresponding results when our proposed V-based method is used.

We have two main observations. First, when the same interaction sampling methods are adopted, the introduction of virtual user profile sampling can effectively degrade the recommendation performance. The attack performance is even more significant when user sampling is conducted by colluding malicious users. When Rating sampling is applied, the attack performance is improved to 7.5%. The ColSH sampling further improves IndSH by 3.7%. Second, our combination of V-based interaction sampling with recommendation rating-based sampling improves the degrading performance of our attack. Although the improvement is not impressive when attacking secret-hiding FedNCF, our interaction sampling method can significantly increase the degrading performance in the Filmtrust. Especially, when no user sampling is applied, only our interaction sampling (Single-VRating) can bring about 24.7% in degrading performance of NDCG. From the above analysis, we can see the overwhelming degrading performance of FRecAttack^2 comes from both our user sampling method and interaction sampling method. In other words, user sampling is necessary in untargeted attacks and colluding malicious users can incur more damage. In addition, this also confirms the effectiveness of our proposed switching mechanism between V-based interaction sampling and the previous sampling method based on recommendation ratings.

5.5 Resilience of our attack against defences

To verify the effectiveness of our attacks in circumventing the defence strategies, we focus on mainstream defences widely used in FL and compare attack effectiveness of different attacks when these defences are deployed. The following defences are selected: (1) **Trimmed-mean** (T.M.) [36]: computing the trimmed-mean of each update dimension; (2) **Krum** (Kr.) [3]: selecting the local model updates which are the most similar to other clients' local updates as global model updates; (3) **Multi-krum** (M.Kr.) [3]: selecting multiple local model updates (10 in our experiments) that are the most similar to other updates as final global updates; (4) **Median** (Med.) [36]: selecting the median value of each parameter as global model updates; and (5) **NormBound** (N.B.) [30]: clipping the L_2 norm of gradients with a threshold (2 in our experiments).

We take to show the impact of these defences on FRecAttack^2 with and without colluding malicious users in Figure 4. As a reference, N.D. represents the recommendation performance when no defence is employed. We can see that the deployment of defensive mechanisms also degrades the recommendation performances. On average, without any attacks, the recommendation performance of FedSoG degrades by about 1.64%. The Krum mechanism compromises the performance the most significantly by about 5.77%. The attack performance of FRecAttack^2 decreases as well after defences are in place. On average, it decreases by 7.49% for HR@10 and 9.85% for NDCG. Comparing the normal decrease, we can see the impacts on our attack are rather marginal. The largest decrease in attack performance occurs when NormBound is applied, and is just about 11.27% when measured by HR@10 and 16.26% when NDCG is used.

To better illustrate the robustness of FRecAttack^2, we further use PCA to visualise the differences in model gradients of benign

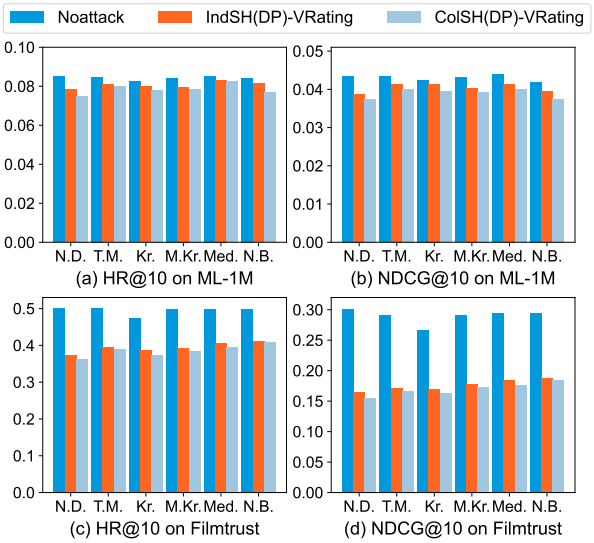

Figure 4: Attack performance under mainstream defences.

users and malicious users. As shown in Figure 5, we noted that the malicious gradients display similar PCA distributions to the majority of benign gradients, while a few gradients from benign users appear as outliers. This indicates that distinguishing malicious users only based on model gradients is challenging and we need a more effective method to detect it.

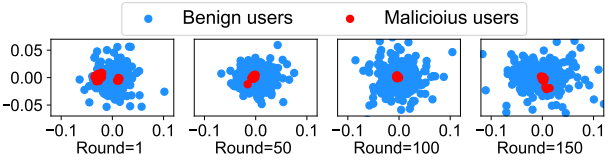

Figure 5: Visualisation of model gradients in different rounds of model training (FedNCF on ML-1M).

5.6 Potential countermeasure

Our FRecAttack[2] also reveals that the interplay between items and user profiles is vital for protecting FR systems. Thus, we further propose a defence mechanism based on contribution quantification (GuardCQ) that detects malicious users by quantifying the contribution of different users to the right interplay between items and user profiles. Specifically, we define "right interplay" as rating behaviours consistent with the global model and select popular items, which are most likely to be exploited by malicious users for attacks, to quantify the contributions. Unpopular items are not considered due to their low prediction accuracy (as observed in Figure 2).

To justify the feasibility of GuardCQ, we use the ML-1M and Filmtrust datasets, showing the difference of consistency in rating behaviours of malicious and benign users on real popular items under attack conditions. Higher consistency indicates a greater contribution to the right interplay. As shown in Figure 6, the blue box and red box represent benign and malicious users, respectively. With model training, the rating behaviours of benign users on popular items are increasingly consistent with the global model, while malicious users exhibit an opposite trend. Our GuardCQ mechanism aims to identify potential malicious users through this contribution difference. As it does not require new information than those already used in the original FR systems, its deployment will not further compromise users' privacy.

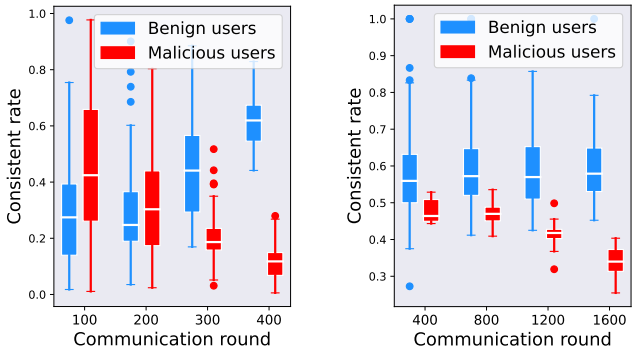

Figure 6: Rating behaviour of benign users and malicious users in different rounds of model training.

5.6.1 Our GuardCQ *Defence.* To implement GuardCQ, three problems need to be considered: 1) How to infer popular items; 2) How to capture users' contributions to the right interplay; 3) How to identify potential malicious users based on contribution differences.

Inferring popular item. Just as discussed in FRecAttack[2], the popular items can be determined through both absolute score and velocity-based ways. We employ the same approach for the aggregator. Moreover, the aggregator is capable of identifying popular by counting the frequency of interactions reported by each selected user $u \in \overline{\mathcal{U}}$ in each round. The sets of popular items obtained by these three ways are denoted as $\mathcal{D}_{AS}, \mathcal{D}_{VB}, \mathcal{D}_{RI}$, respectively.

Quantifying users' contributions. Considering the occasional randomness in single rating behaviours, we adopt the "multi-round comprehensive assessment" principle, where a user's contribution value ctb_u in the current round is jointly determined by its recent w' rating behaviours. In Alg. 7, we show the corresponding pseudocode. For each popular item set obtained above, we assign two sets, F_u and $\hat{F}_u$, to each user for recording his/her rating behaviours in the last w' rounds (line 3–7). Taking the example of user u and set $\mathcal{D}_{AS}$, $F_{u_{AS}}$ records the count n_{AS} of the intersection $\mathcal{D}_{u \cap AS}$ between the user's interaction items $\mathcal{D}_u$ and the predicted popular items $\mathcal{D}_{AS}$. $\hat{F}_{u_{AS}}$ records the count of consistent rating behaviours between the user u and the global model on items $i \in \mathcal{D}_{u \cap AS}$. Now, the contribution of user u to the right interplay in the ℓ-th round can be expressed as: $ctb_u = \frac{\sum_{i=\ell-w'}^{\ell} \hat{F}^{\ell}_{u_{AS}}(i)}{\sum_{i=\ell-w'}^{\ell} F^{\ell}_{u_{AS}}(i)}$. $CTB_{AS} = \{ctb_u | u \in \overline{\mathcal{U}}\}$ is composed of the contributions from all users participating in the ℓ-th round of training (line 13–17). To reduce false positives of malicious users, we only perform the above operation for users whose intersection count exceeds a certain threshold, i.e.,*i.e.*, $|\mathcal{D}_u| > \tau$.

Algorithm 7: Quantifying contributions for GuardCQ

Function SignConsistRatio($\mathbf{Y}^\ell, \mathbf{\Theta}^\ell_\psi, \mathcal{D}_{\overline{\mathcal{U}}}, \lambda, \mathcal{D}^\ell_{AS}, \mathcal{D}^\ell_{VB}, \mathcal{D}^\ell_{RI}$):
  $CTB_{AS} = \{\}, CTB_{VB} = \{\}, CTB_{RI} = \{\}$
  foreach $u \in \overline{\mathcal{U}}$ **do**
    foreach $q \in \{AS, VB, RI\}$ **do**
      $F^\ell_{u_q} = \{\}, \quad \hat{F}^\ell_{u_q} = \{\}$
      $\mathcal{D}^\ell_{u\cap q} \leftarrow \mathcal{D}_u \cap \mathcal{D}^\ell_q, \quad n^\ell_q \leftarrow |\mathcal{D}^\ell_{u\cap q}|$
    end
    if $\mathcal{D}_u > \tau$ **then**
      $\check{\mathbf{Y}}^{\ell+1}_u \leftarrow \mathbf{Y}^\ell - \lambda \nabla \mathbf{Y}^\ell_u$
      $\check{\mathbf{\Theta}}^{\ell+1}_u \leftarrow \mathbf{\Theta}^\ell - \lambda \nabla \mathbf{\Theta}^\ell_u$
      $\Delta\mathcal{R}_u \leftarrow \sum_{\ddot{u}\in\ddot{\mathcal{U}}} \psi(\mathbf{x}_{\ddot{u}}, \check{\mathbf{Y}}^{\ell+1}_u) - \sum_{\ddot{u}\in\ddot{\mathcal{U}}} \psi(\mathbf{x}_{\ddot{u}}, \mathbf{Y}^\ell_u)$
      $s_u \leftarrow \{\frac{\mathrm{sgn}(\Delta\mathcal{R}^i_u)+1}{2} \mid i \le |\mathcal{I}|\}$
      foreach $q \in \{AS, VB, RI\}$ **do**
        $F^\ell_{u_q} \leftarrow F^{\ell-1}_{u_q} \cup \{n^\ell_q\}$
        $\hat{F}^\ell_{u_q} \leftarrow \hat{F}^{\ell-1}_q \cup \{\sum_{i=1}^{n^\ell_q} s_u(\mathcal{D}^\ell_{u\cap q})\}$
        $CTB_q \leftarrow CTB_q \cup \{\frac{\sum^\ell_{i=\ell-w'} \hat{F}^\ell_{u_q}(i)}{\sum^\ell_{i=\ell-w'} F^\ell_{u_q}(i)}\}$
      end
    end
  end
  return CTB_{AS}, CTB_{VB}, CTB_{RI}

Detecting malicious users. Due to different optimisation objectives, there inevitably exists a discrepancy in the contributions to the "right interplay" between malicious and benign users. As shown in Figure 6, the contribution *ctb* from benign users gradually increases and approaches 1.0, while the contribution from malicious users decreases and approaches zero. This difference will become more obvious and form a clear *gap* as the model training progresses. Our objective is to detect malicious users by pinpointing the maximum gap between CTB in each round. Specifically, we sort the CTB to obtain CTB', then calculate the maximum difference between adjacent values within CTB', i.e.,$max(CTB'(i+1) - CTB'(i), i \le |CTB'|)$. The $CTB'(i+1)$ corresponding to the maximum gap is considered the minimum contribution value ctb^* that benign users should satisfy. In this way, users with a contribution less than ctb^* are temporarily labelled as malicious. This process is applied to $CTB_{AS}, CTB_{VB}, CTB_{RI}$, and a user is finally labelled as malicious only when identified as such in all three situations.

5.6.2 Defence Performance Evaluation. Table 5 shows the results after using the GuardCQ mechanism on the Filmtrust dataset against different attacks. Just the same as prior defences, it is inevitable that GuardCQ also generates a slight impact on the model's recommendation performance. This is because GuardCQ may mistakenly filter out extremely abnormal gradients from benign users. Secondly, GuardCQ effectively alleviates the impact of malicious users on the recommendation model. For example, on the Filmtrust dataset, deploying GuardCQ decreases the attack performance of FRecAttack^2-ColDP from 27.56% to 8.78%.

It is noted that GuardCQ can also be integrated with existing defences mentioned above in Section 5.5 as a screening mechanism to filter out suspicious users. We tested the combination of GuardCQ with other mechanisms like NormBound and Trimmedmean, which have minimal impact on model performance. The result shown in Table 6 demonstrate that combining GuardCQ enhances the resistance of existing defences against untargeted poisoning attacks in FR systems. On the Filmtrust dataset, deploying GuardCQ+N.B. further reduces the attack performance of FRecAttack^2-ColDP from 8.78% to 3.29%.

Table 5: Defence performance of GuardCQ against different attacks with 10% malicious clients on Filmtrust (K=10).

Filmtrust	NoDefense		GuardCQ	
	HR@10	NDCG@10	HR@10	NDCG@10
NoAttack	0.5022	0.3003	0.4987	0.2966
SignFlip	0.4087	0.1849	0.4825	0.2799
LabelFlip	0.4984	0.2786	0.4957	0.2914
FedAttack	0.4066	0.1843	0.4994	0.2964
Gaussian	0.4835	0.2550	0.5034	0.3009
LIE	0.4797	0.2380	0.5005	0.2948
FRecAttack^2-IndDP	0.3744	0.1644	0.4706	0.2678
FRecAttack^2-ColDP	0.3638	0.1553	0.4581	0.2637

Table 6: Defence performance of GuardCQ+X on Filmtrust.

	FRecAttack^2-IndDP		FRecAttack^2-ColDP	
	HR@10	NDCG@10	HR@10	NDCG@10
NoDefense	0.3744	0.1644	0.3638	0.1553
GuardCQ+N.B.	0.4865	0.2679	0.4875	0.2643
GuardCQ+T.M.	0.4952	0.2820	0.4949	0.2818

6 CONCLUSION AND FUTURE WORK

In this study, we proposed the first formal framework for untargeted poisoning attacks against FR systems. Our framework unveils the important role of the interplay between user profiles and items in determining the performance of untargeted attacks. We propose a new attack FRecAttack^2 which impairs prior research by comprehensively exploiting the user-item interplay. Specifically, we presented the implementation of two crucial components in untargeted attacks: virtual user sampling and interaction sampling. Meanwhile, we proposed a new defence mechanism called GuardCQ to mitigate the impact of untargeted attacks. It detect malicious users by quantifying their contribution to the right interplay between items and user profiles. Through extensive empirical evaluation, we successfully illustrated the effectiveness of our FRecAttack^2 attack, which outperforms prior untargeted attacks even with a small number of malicious users, and our GuardCQ defence, which provides more comprehensive protection for FR systems.

ACKNOWLEDGEMENTS

This research was funded in whole, or in part, by the Systematic Major Project of China State Railway Group Co. Ltd., under Grant P2023W002, and the Luxembourg National Research Fund (FNR), grant reference C21/IS/16281848 (HETERS).

REFERENCES

[1] Muhammad Ammad-Ud-Din, Elena Ivannikova, Suleiman A Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. 2019. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888* (2019).

[2] Gilad Baruch, Moran Baruch, and Yoav Goldberg. 2019. A Little Is Enough: Circumventing Defenses For Distributed Learning. In *the 2019 Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc., 8632–8642.

[3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *the 2017 Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., 119–129.

[4] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Gong. 2021. FLTrust: Byzantine-robust federated learning via trust bootstrapping. In *the 28th Annual Network and Distributed System Security Symposium*. The Internet Society.

[5] Xiaoyu Cao, Jinyuan Jia, Zaixi Zhang, and Neil Zhenqiang Gong. 2022. FedRecover: Recovering from Poisoning Attacks in Federated Learning using Historical Information. In *the 2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 326–343.

[6] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2021. Secure Federated Matrix Factorization. *IEEE Intelligent Systems* 36, 5 (2021), 11–20.

[7] Yudong Chen, Lili Su, and Jiaming Xu. 2018. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. In *the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*. ACM, 96.

[8] Ziqian Chen, Fei Sun andYifan Tang, Haokun Chen, Jinyang Gao, and Bolin Ding. 2023. Studying the Impact of Data Disclosure Mechanism in Recommender Systems via Simulation. *ACM Transactions on Information Systems* 41, 3 (2023), 60:1–60:26.

[9] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.

[10] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *the 29th USENIX Security Symposium*. USENIX Association, 1605–1622.

[11] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems* 1, 1 (2023), 1–51.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *the 26th International Conference on World Wide Web*. ACM, 173–182.

[13] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *the 2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 19–35.

[14] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. In *the 2016 NIPS Workshop on Private Multi-Party Machine Learning*.

[15] Haoyang LI, Qingqing Ye, Haibo Hu, Jin Li, Leixia Wang, Chengfang Fang, and Jie Shi. 2023. 3DFed: Adaptive and Extensible Framework for Covert Backdoor Attack in Federated Learning. In *the 2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 1893–1907.

[16] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. 2019. RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *the 33rd AAAI Conference on Artificial Intelligence*, Vol. 33. AAAI Press, 1544–1551.

[17] Feng Liang, Weike Pan, and Zhong Ming. 2021. Fedrec++: Lossless federated recommendation with explicit feedback. In *the 35th AAAI conference on artificial intelligence*, Vol. 35. AAAI Press, 4224–4231.

[18] Guanyu Lin, Feng Liang, Weike Pan, and Zhong Ming. 2020. Fedrec: Federated recommendation with explicit feedback. *IEEE Intelligent Systems* 36, 5 (2020), 21–30.

[19] Zhiwei Liu, Liangwei Yang, Ziwei Fan, Hao Peng, and Philip S Yu. 2022. Federated social recommendation with graph neural network. *ACM Transactions on Intelligent Systems and Technology* 13, 4 (2022), 1–24.

[20] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–136.

[21] Xiaoting Lyu, Yufei Han, Wei Wang, Jingkai Liu, Bin Wang, Jiqiang Liu, and Xiangliang Zhang. 2023. Poisoning with cerberus: stealthy and colluded backdoor attack against federated learning. In *the 37th AAAI Conference on Artificial Intelligence*. AAAI Press.

[22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *the 20th International Conference on Artificial Intelligence and Statistics (Machine Learning Research, Vol. 54)*. PMLR, 1273–1282.

[23] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting Unintended Feature Leakage in Collaborative Learning. In *the 2019 IEEE Symposium on Security and Privacy*. IEEE, 691–706.

[24] Vasileios Perifanis and Pavlos S. Efraimidis. 2022. Federated Neural Collaborative Filtering. *Knowledge-Based Systems* 242 (2022), 108441.

[25] Dazhong Rong, Qinming He, and Jianhai Chen. 2022. Poisoning deep learning based recommender model in federated learning scenarios. In *the 31st International Joint Conference on Artificial Intelligence*. ijcai.org, 2204–2210.

[26] Dazhong Rong, Shuai Ye, Ruoyan Zhao, Hon Ning Yuen, Jianhai Chen, and Qinming He. 2022. FedRecAttack: Model poisoning attack to federated recommendation. In *the 38th IEEE International Conference on Data Engineering*. IEEE, 2643–2655.

[27] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. 2022. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *the 2022 IEEE Symposium on Security and Privacy*. IEEE, 1354–1371.

[28] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daumé III, and Tudor Dumitras. 2018. When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks. In *the 27th USENIX Security Symposium*. USENIX Association, 1299–1316.

[29] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data Poisoning Attacks Against Federated Learning Systems. In *the 25th European Symposium on Research in Computer Security (Lecture Notes in Computer Science, Vol. 12308)*. Springer, 480–501.

[30] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput andHarit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris S. Papailiopoulos. 2020. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In *the 2020 Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 16070–16084.

[31] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. FedGNN: Federated Graph Neural Network for Privacy-Preserving Recommendation. In *the 2021 ICML Workshop on Federated Learning for User Privacy and Data Confidentiality*. PMLR.

[32] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2022. FedAttack: Effective and covert poisoning attack on federated recommendation via hard sampling. In *the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 4164–4172.

[33] Hong Xuan, Abby Stylianou, Xiaotong Liu, and Robert Pless. 2020. Hard Negative Examples are Hard, but Useful. In *the 16th European Conference on Computer Vision (Lecture Notes in Computer Science, Vol. 12359)*. Springer, 126–142.

[34] Liu Yang, Junxue Zhang, Di Chai, Leye Wang, Kun Guo, Kai Chen, and Qiang Yang. 2022. Practical and Secure Federated Recommendation with Personalized Mask. In *the 2022 IJCAI workshop on Trustworthy Federated Learning (Lecture Notes in Computer Science, Vol. 13448)*. Springer, 33–45.

[35] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology* 10, 2 (2019), 12:1–12:19.

[36] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett. 2018. Byzantine-robust distributed learning: towards optimal statistical rates. In *the 35th International Conference on Machine Learning (Machine Learning Research, Vol. 80)*. PMLR, 5650–5659.

[37] Yang Yu, Qi Liu, Likang Wu, Runlong Yu, Sanshi Lei Yu, and Zaixi Zhang. 2023. Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense. In *the 37th AAAI Conference on Artificial Intelligence*, Vol. 37. AAAI Press, 4854–4863.

[38] Wei Yuan, Quoc Viet Hung Nguyen, Tieke He, Liang Chen, and Hongzhi Yin. 2023. Manipulating Federated Recommender Systems: Poisoning with Synthetic Users and Its Countermeasures. In *the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1690–1699.

[39] Wei Yuan, Chaoqun Yang, Quoc Viet Hung Nguyen, Lizhen Cui, Tieke He, and Hongzhi Yin. 2023. Interaction-level Membership Inference Attack Against Federated Recommender Systems. In *the 2023 ACM Web Conference*. ACM, 1053–1062.

[40] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Quoc Viet Hung Nguyen, and Lizhen Cui. 2022. PipAttack: Poisoning federated recommender systems for manipulating item promotion. In *the 15th ACM International Conference on Web Search and Data Mining*. ACM, 1415–1423.