FedAttack: Effective and Covert Poisoning Attack on Federated Recommendation via Hard Sampling

Chuhan Wu[1], Fangzhao Wu[2], Tao Qi[1], Yongfeng Huang[1], Xing Xie[2]
[1]Department of Electronic Engineering, Tsinghua University, Beijing 100084
[2]Microsoft Research Asia, Beijing 100080, China
{wuchuhan15,wufangzhao,taoqi.qt}@gmail.com,yfhuang@tsinghua.edu.cn,xingx@microsoft.com

ABSTRACT

Federated learning (FL) is a feasible technique to learn personalized recommendation models from decentralized user data. Unfortunately, federated recommender systems are vulnerable to poisoning attacks by malicious clients. Existing recommender system poisoning methods mainly focus on promoting the recommendation chances of target items due to financial incentives. In fact, in real-world scenarios, the attacker may also attempt to degrade the overall performance of recommender systems. However, existing general FL poisoning methods for degrading model performance are either ineffective or not concealed in poisoning federated recommender systems. In this paper, we propose a simple yet effective and covert poisoning attack method on federated recommendation, named *FedAttack*. Its core idea is using globally hardest samples to subvert model training. More specifically, the malicious clients first infer user embeddings based on local user profiles. Next, they choose the candidate items that are most relevant to the user embeddings as hardest negative samples, and find the candidates farthest from the user embeddings as hardest positive samples. The model gradients inferred from these poisoned samples are then uploaded to the server for aggregation and model update. Since the behaviors of malicious clients are somewhat similar to users with diverse interests, they cannot be effectively distinguished from normal clients by the server. Extensive experiments on two benchmark datasets show that *FedAttack* can effectively degrade the performance of various federated recommender systems, meanwhile cannot be effectively detected nor defended by many existing methods.

KEYWORDS

Federated learning, Recommendation, Poisoning attack, Hard sampling

ACM Reference Format:
Chuhan Wu[1], Fangzhao Wu[2], Tao Qi[1], Yongfeng Huang[1], Xing Xie[2]. 2022. FedAttack: Effective and Covert Poisoning Attack on Federated Recommendation via Hard Sampling. In *Proceedings of the 28th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington D.C.*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

KDD '22, August 14–18, 2022, Washington D.C.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

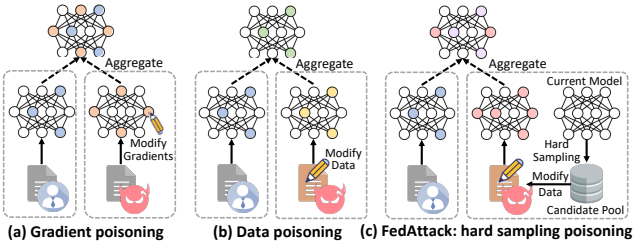


Figure 1: Existing popular poisoning paradigms on FL and our *FedAttack* for poisoning federated recommendation.

1 INTRODUCTION

Personalized recommendation techniques are widely used in many online scenarios such as e-commerce [16] and news feed [47] to alleviate users' information overload. Conventional recommender systems are learned on centralized user data, which may arouse considerable privacy concerns and risks [54]. Federated learning (FL) [26] is a privacy-preserving paradigm that enables learning models on decentralized data under privacy protection [55]. It has been successfully applied to the recommendation field to train privacy-aware recommender systems without collecting and exchanging raw user data [19, 20, 48], where user privacy can be protected to a certain extent.

Unfortunately, due to the decentralization of model training, not all clients in federated learning are trusted and a fraction of them may be controlled by malicious parties [43]. Thus, federated learning-based systems are more vulnerable to poisoning attack than centralized learning [1]. In federated recommendation, most prior poisoning works focus on promoting target items by increasing their exposure chances [60], which is usually motivated by financial incentives [6]. Besides item promotion, the attacker may aim to degrade the overall performance of federated recommender systems [22], which may be due to the cut-throat competitions among companies [24]. This attack form is known as untargeted attack [23]. In this paper, we study untargeted poisoning attack on federated recommendation, which is a less-explored problem.

There are several methods for untargeted poisoning attack on federated learning in general domains [2, 10]. One way is poisoning model gradients, as shown in Fig. 1(a). A naive method is adding strong noise to the local model gradients uploaded to the server for aggregation. However, the outlier gradients can be easily recognized by anomaly detection methods [28] or defended by robust gradient aggregation methods [4]. Thus, a few studies explore more covert ways to poison the model gradients. For example, Baruch

et al. [2] proposed to add a small amount of noise to each dimension of the average model updates from benign clients, where the noise intensity is estimated by the number of benign and Byzantine clients. Fang et al. [10] proposed to perturb the gradients by adding noise with opposite parameter update directions inferred from benign updates. However, these methods usually require a large fraction of Byzantine workers (e.g., 20%) to achieve a notable performance decrease, which is quite expensive when there are a large number of clients in FL [36]. In addition, they need strong assumptions on the attackers' capability such as background knowledge of benign clients' updates and the server's aggregation rule, which may be somewhat difficult to obtain in realistic attacking scenarios. Another possible way for untargeted attack is data poisoning, as shown in Fig. 1(b). Most prior studies on data poisoning are conducted in centralized settings [34]. Some methods aim to generate adversarial samples [7, 27, 53], which usually require strong background knowledge of training data. However, they are not applicable in federated learning because the attackers cannot access other clients' data. Label flipping is a simple data poisoning method without prior knowledge of training data [51]. For example, Fang et al. [10] showed the possibility of poisoning federated learning models by changing true labels to false labels on each Byzantine client. However, label flipping is also ineffective when the ratio of controlled clients is insufficient [36].

Different from poisoning general FL systems, untargeted poisoning attack on federated recommendation faces several unique challenges. First, the number of clients participating in federated recommendation can be rather large, and it can be extremely expensive to control a significant portion of clients [30]. Thus, the attacking algorithm needs to be effective under limited Byzantine workers. Second, many recommendation models are learned on implicit user feedback that contains heavy noise [45], and thereby the models are robust to perturbation to some extent [57]. Thus, noise injecting and label flipping operations may have very limited impacts on final recommendation performance. Third, since local user data usually only involves a small subset of items, the updates of local parameters, especially item embeddings, can be very sparse [3]. Therefore, aggressive modification on local model updates may frequently trigger anomaly detection and defense mechanisms on the aggregation server.

To address the above challenges, in this paper, we propose a both effective and covert poisoning attack method named *FedAttack*[1], which aims to degrade the performance of federated recommender systems with a small fraction of malicious clients. The core idea of our approach is inspired by hard negative sampling [15], which is a widely used technique in information retrieval to find informative negative samples. Although hard samples are beneficial for learning more discriminative models [33], globally hardest negative examples are usually harmful to model performance because many of them are false negatives and they can lead to bad local minima early in training [52]. Thus, in our work we propose to use globally hardest sampling as a poisoning technique, as shown in Fig. 1(c). More specifically, on each Byzantine client we first use the current model to infer the user embedding from local user profiles. Next, each Byzantine client uses its local user embedding to retrieve globally hardest negative samples with the closest distance to the user embedding from the candidate pool. To further increase the difficulty of model training, we also retrieve pseudo "hardest positive samples" that are farthest from user embeddings to replace the original positive samples. By training recommendation models on these hard samples, the Byzantine clients can finally generate poisoned gradients to upload. These gradients have significant impacts on model convergence, while hard for the server to perceive because they are similar to the gradients generated by benign users with strong preferences on the diversity of recommendation results [49].[2] Extensive experiments on two benchmark recommendation datasets show that *FedAttack* can effectively degrade the performance of various federated recommendation models with a small fraction of Byzantine clients and can circumvent many defense mechanisms.

The contributions of this paper are listed as follows:

- To our best knowledge, our approach is the first untargeted poisoning attack method on federated recommendation. It reveals the security risk of recommenders even under decentralized settings with a small fraction of malicious clients, and can raise researchers' attention on enhancing the security of federated recommender systems.
- We are the first to employ hard sampling mechanism as an attack technique for poisoning federated recommendation. We demonstrate that globally hardest samples can subvert the model training in an effective and covert way.
- We conduct extensive experiments on two benchmark datasets, validating that *FedAttack* is more effective than many baseline methods in attacking and defense circumventing.

2 RELATED WORK

The security issues in machine learning systems have been extensively studied [9], and poisoning attack is an emerging threat in recent years [36]. Data poisoning is one of the most common forms of poisoning attack [38]. With a fraction of designed points injected into the training data, the malicious attacker can degrade the model performance in an indiscriminate or targeted way [27]. Label flipping is a simple and widely used method without the need of background knowledge of training data [50]. For example, Xiao et al. [51] proposed a label flipping method that aims to change true sample labels into false ones which can maximize the classification error. However, these carefully designed samples can be identified as outliers and filtered [41]. Another popular data poisoning way is generating adversarial samples to subvert the training process [44]. For example, Yang et al. [53] proposed to use autoencoder as the generator to create poisoned data, which is further updated based on a reward function of loss. The target model serves as the discriminator that calculates the loss of discriminating between poisoned data and normal data. Muñoz-González et al. [27] proposed a similar generative adversarial network based data poisoning method that uses a generator to generate poisoning samples which can maximize classification errors in the target task and meanwhile minimize the discriminator's ability in distinguishing poisoned data from normal data. These data poisoning methods are conducted

[1]Source code is available at https://github.com/wuch15/FedAttack

[2]These users may interact with items that are diverse from previously interacted ones, which have similar effects on model learning with hard sampling.

on centralized datasets, where the attacker has strong prior knowledge of the training dataset [8]. Thus, they are not applicable when training data is decentralized and cannot be exchanged.

In recent years, several works study the poisoning attack problem in general federated learning [5]. According to the attack goals, they can be classified into targeted attack (misclassify specific subsets of inputs), backdoor attack (misclassify inputs with certain patterns), and untargeted attack (degrading overall performance) [36]. Here we focus on untargeted attack that is most relevant to our work. Some centralized data poisoning methods can be directly applied to FL poisoning. For instance, Fang et al. [10] showed the potential of label flipping in poisoning attack on FL by simply replacing true labels with other false ones. Different from data poisoning in centralized learning, in federated learning the attacker can directly manipulate the model updates to achieve stronger attack performance. For example, Baruch et al. [2] proposed to first estimate the standard deviation of benign model gradients, then compute an intensity coefficient based on the number of benign and Byzantine clients, and finally modify the model gradients by adding noise according to the standard deviation and intensity coefficient. Fang et al. [10] proposed to perturb the gradients by adding noise with constant values but opposite signs with the benign model updates. The attacker needs to empirically select the noise constant that can effectively circumvent the target defense method. Shejwalkar and Houmansadr [35] further proposed an improved attacking method by using dynamic and data-dependent malicious direction to perturb the model updates. However, these methods usually require a large fraction of Byzantine clients for effective attack, which may lead to heavy costs when there are a large number clients in FL.

Different from poisoning attack on general classification models, recommender system poisoning methods may aim to promote a set of target items due to financial incentives [7, 37, 46, 58, 59]. For example, Li et al. [18] proposed to generate malicious samples that optimize the tradeoff between maximizing overall benign recommendation errors and promoting ratings of target items. Fang et al. [11] proposed a graph-based recommender system poisoning method that finds optimal edge weights between target items and users as well as assigns maximum ratings to them. They further added filler items with the largest edge weights and generated random ratings for them to mimic normal users. These methods are aimed at centralized recommendation model learning, where attackers usually have full knowledge of the training data [42]. In federated recommendation scenarios, the attacker cannot access the data on normal clients, and there are very few methods for federated recommendation poisoning. PipAttack [60] is a recent item promotion method that promotes the ranking scores and estimated popularity of target items, meanwhile uses a distance constraint that regularizes the distance between poisoned and original gradients. However, there still lacks research of untargeted poisoning attack on federated recommendation, which has many distinct challenges from general FL poisoning. Our approach is the first one for untargeted federated recommendation poisoning, which can achieve more effective and covert attacks than many general FL poisoning methods. It reveals a new form of security risk in federated recommendation that needs to be considered in future recommender system design.

3 METHODOLOGY

We introduce the details of our *FedAttack* approach in this section. We first present a definition of the problem studied in this paper, next discuss the prior knowledge and capability of the attacker, then describe the basic recommendation model learning framework on benign clients, and finally introduce how to conduct hard sampling-based poisoning attack on Byzantine clients.

3.1 Problem Definition

Assume there are N clients in total for federated recommendation model learning, where $M\%$ of them are Byzantine clients controlled by a malicious attacker. Each client keeps a local copy of the recommendation model, which is collaboratively learned and updated (note that there may be some personalized parameters on each client like user embeddings). These clients are all coordinated by a server that receives local model updates from clients and aggregates them into a global update to deliver. Each client locally maintains a user profile (user ID or historical interacted items) that is used to infer user interest. On each client, the recommendation model aims to predict a ranking score for each candidate item that indicates the probability of this user interacting with this item. The items in the candidate set are ranked based on their ranking scores. The Byzantine clients controlled by the attacker aim to generate poisoned gradients that can effectively degrade the recommendation performance on benign clients, meanwhile cannot be easily detected nor defended by the aggregation server.

3.2 Assumption and Prior Knowledge

In *FedAttack*, we assume that the parameter set of the recommendation model includes a user model for inferring user interest embedding from user profiles, an item model that converts item IDs into item embeddings, and a predictor model that computes personalized ranking scores based on the relevance between user interest embeddings and item embeddings. During the model training, all benign clients only upload their model updates to the server, and Byzantine clients cannot access their raw data or intermediate model results due to privacy reasons. In addition, the attacker does not know the aggregation rules on the server. To conduct the attack, we assume that the attacker has the following capability and prior knowledge: (1) the Byzantine clients have normal user profiles, which can be achieved by hacking normal clients; (2) the attacker knows a large candidate item pool (would be better if it is the whole item set as [60]), which is achievable because the attacker can crawl the publicly available items on e-commerce platforms; (3) the Byzantine clients can modify their local model inputs and labels, and have access to their local user and item embeddings (if not satisfied, the attacker can learn a surrogate model to mimic the original model [61]). In our assumptions, the attacker has much weaker capability and background knowledge of the training data than most existing works on centralized or FL poisoning [36]. Thus, our attack method is easier to execute in real-world scenarios.

3.3 FedAttack Framework

The framework of *FedAttack* is shown in Fig. 2. The Byzantine clients only modify the input samples and their labels, while the model gradients are directly computed on these samples without

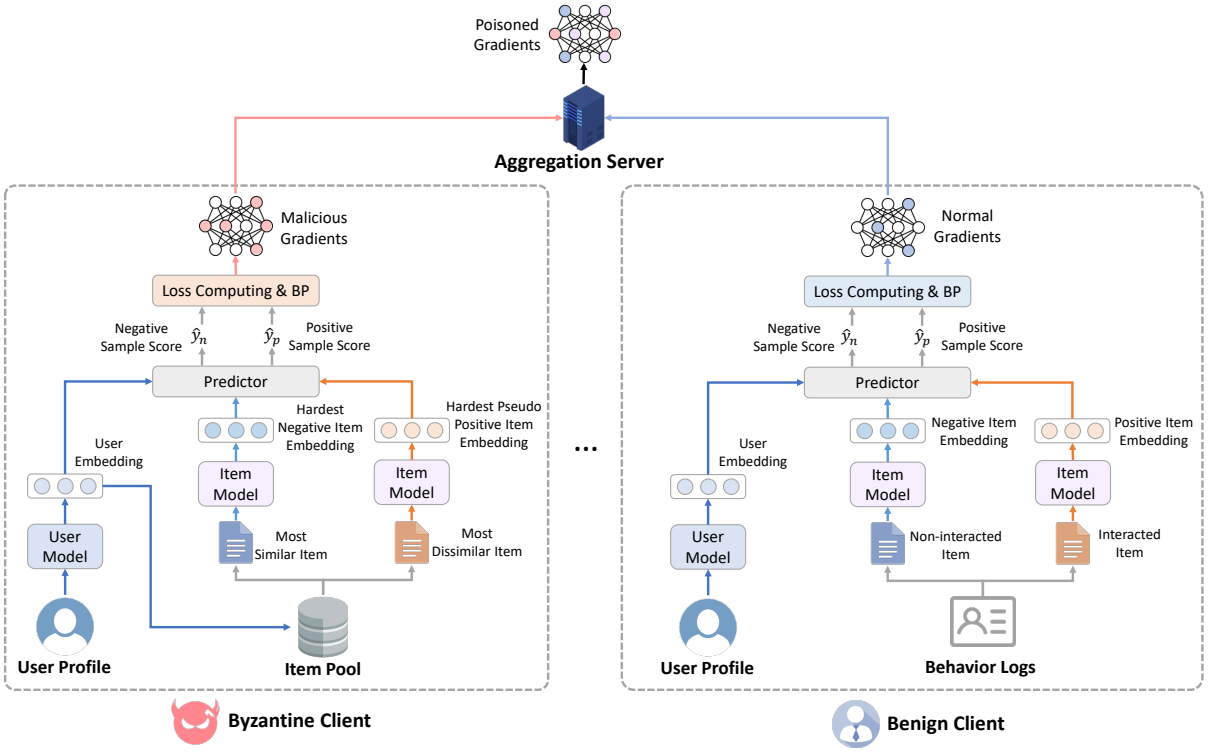


Figure 2: The framework of our *FedAttack* approach.

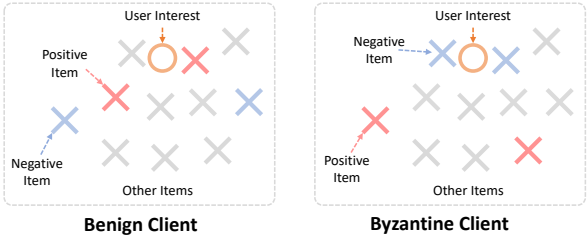


Figure 3: Schematic comparison between negative and positive samples on benign and Byzantine clients.

further manipulation. In each round of model update, the Byzantine clients first use the current user model to infer their user embeddings (denoted as $\mathbf{u}$) from their local user profiles. The user model can be implemented by various architectures, such as the simple user ID embedding in collaborative filter [13], or sequential models like GRU [14] and Transformer [39] that learn user embeddings from historical interacted items. The user embedding is then used to retrieve hard samples on each Byzantine client. Assume there are K original positive samples on a Byzantine client (can be constructed from the interaction history). We take BPR [32] as an example training scheme to explain the poisoning method as follows.

To measure the "hardness" of candidate items, we use the inner product between the user embedding $\mathbf{u}$ and embeddings of candidate items in the item pool as the measurement. We construct an ANN index based on the current item embedding table to achieve efficient candidate item selection. Motivated by the hard negative sampling techniques [15], we use the top K items with the highest similarities to user embeddings as the globally hardest negative samples. These negative samples usually have very high relevance to user interest, though they are not interacted by the user because they may not be exposed. Thus, these samples can be false negatives in recommendation model training and may have huge impacts on model convergence. To subvert the model learning more effectively, we further propose to replace the original positive samples with pseudo ones obtained by hard sampling. We regard the K items with the most dissimilar embeddings to the user embedding as the hardest pseudo positive items. It is usually difficult for the model to predict high ranking scores for these items, and enforcing the model to match them with dissimilar user interest can greatly impair its ability in modeling the relevance between items and user interest. A schematic comparison between positive and negative samples on Byzantine clients and benign clients is shown in Fig. 3. On many benign clients, the positive items are usually similar to user interest, though they may not be the most similar ones. The negative items are usually randomly selected from local data, which are expected to have low similarities with user interest. By contrast, on Byzantine clients the negative items are very similar to the user interest, while positive samples are quite diverse from user interest. Thus, the training samples on Byzantine clients can damage the modeling of user-item relevance, and the poison collaborative signals will propagate in the model training to further affect benign clients. In addition, the patterns of positive items on Byzantine clients have

some similarities to benign users. For example, if a user has only a few historical interacted items with diverse characteristics, the interacted item may have low similarity to the user interest inferred from previously interacted items [29]. Thus, it may be difficult for the server to distinguish Byzantine clients from outlier and cold users (see further discussions in experiments).

After obtaining the hardest positive and negative items, we use the item model to encode their embeddings. We randomly pair the K negative and pseudo positive items to form K positive-negative sample pairs. We denote the embedding of a pair of positive and negative items as $\mathbf{i}_p$ and $\mathbf{i}_n$. A predictor model takes the user embedding and these item embeddings as the input, and calculates the personalized ranking scores for them (independently computed for positive and negative samples). It can also be implemented by various architectures, such as inner product [39] and neural collaborative filtering [13] modules. The predictor outputs a negative sample score $\hat{y}_n$ and a positive sample score $\hat{y}_p$, which are further used to compute the BPR loss $\mathcal{L}$ as follows:

$$\mathcal{L} = -\log(\sigma(\hat{y}_p - \hat{y}_n)), \tag{1}$$

where σ is the sigmoid function. By optimizing the loss function locally, the malicious clients further compute the malicious gradients by backward propagation. Both the malicious gradients on Byzantine clients and normal gradients on benign clients are uploaded to the server for aggregation. The server further distributes the aggregated gradients to each client to conduct local update. This process is repeated until the model converges.

Our approach addresses the three challenges mentioned in the Introduction Section in the following ways. First, since globally hardest samples usually have huge impacts on model convergence, the attack can still be effective when there are very limited Byzantine clients. Second, although the recommendation model may be robust to random noise, it is still vulnerable to our approach because the influence of hard samples is usually accumulated rather than counteracted as noise. Third, different from many gradient poisoning methods that perturb the entire set of items, our approach only modifies the input samples and their labels. The local embedding gradients of involved hard positive and negative items are manipulated, while the rest item embeddings are not updated. The local model updates on Byzantine clients, especially the item embeddings, are still sparse, making it difficult for the server to distinguish between Byzantine and benign clients based on the sparseness of gradients. Therefore, *FedAttack* can attack federated recommendation effectively and covertly.

4 EXPERIMENTS

4.1 Datasets and Experimental Settings

We use two widely used benchmark experiments for recommendation in our experiments. The first dataset is MovieLens-1M [12], which is a movie recommendation dataset. The second dataset is the Amazon Beauty dataset [25] (denoted as Beauty). It is a domain subset of the amazon dump dataset that can be used for e-commerce recommendation. The statistics of the two datasets are shown in Table 1. Following [39], we use the last interacted item of each user as the test sample, the item before the last one for validation, and the rest items for construct training samples.

Table 1: Statistics of the two datasets.

Datasets	#Users	#Items	#Actions	Avg. length	Density
MovieLens-1M	6,040	3,416	1.0m	163.5	4.79%
Beauty	40,226	54,542	0.35m	8.8	0.02%

In our experiments, we choose neural collaborative filtering (NCF) [13] and BERT4Rec [39] models as the basic victim models for attack. We use the standard federated learning framework with Adam optimizer [31] for model training, and the learning rate is 1e-3. The hidden dimension of models is 64. The number of clients to form a batch is 16. The maximum epoch is 50. We randomly sample $M\%$ ($M \in \{1, 2, 5\}$) of clients as Byzantine clients. The negative samples on benign clients are randomly drawn from the whole item set. The Hit Ratio (HR) and normalized Discounted Cumulative Gain (nDCG) over the top 5 ranked items are used as the recommendation performance metrics. Note that the metric is only calculated on benign clients. Since some recent works figured out that sampled evaluation is biased [17], we evaluate the recommendation performance by ranking the entire item set rather than the sampled subsets based on popularity [39]. Each experiment is repeated 5 times and the average results are reported.

4.2 Attack Performance Evaluation

We compare the attack performance of *FedAttack* with several untargeted poisoning methods on federated learning, including: (1) *LabelFlip* [10], flip the positive and negative item labels on Byzantine clients; (2) *Gaussian* [10], using Gaussian noise that has the same distribution with the before-attack local models to replace Byzantine clients' local models; (3) *LIE* [2], adding small amounts of noise to the average of benign gradients; (4) *STAT-OPT* [10], adding constant noise with opposite directions to the average of benign gradients; (5) *DYN-OPT* [35], adding noise with dynamic malicious directions to the average of benign gradients. Since the total number of clients in federated recommender systems is usually large, we limit the ratio of malicious clients to three different small values, i.e., 1%, 2%, and 5%. In addition, we report the model results without attack for benchmarking the attack performance. The results are shown in Table 2, from which we have several findings. First, all compared attack methods can degrade the recommendation performance when there are relatively sufficient Byzantine clients (e.g., 5%), while most baseline methods are ineffective when Byzantine clients are very limited. Especially, the *LabelFlip* and *Gaussian* methods can even slightly raise the performance when there are only 1% malicious clients. This may be because these perturbation methods can make the model more robust to the noise in user behaviors. Second, gradient poisoning methods such as *LIE*, *STAT-OPT*, and *DYN-OPT* have better attack effectiveness than *LabelFlip* and *Gaussian*. This is because the noise introduced by perturbing the labels or the models on a small fraction of clients is weak, which cannot effectively subvert the training process. Third, *FedAttack* has consistently better attack performance than other baselines, and their gaps are significant ($p < 0.01$ in t-test). This is because the hardest positive and negative samples have strong impacts on model convergence, and the poisoned collaborative information can be further propagated to benign clients in model training. In addition,

Table 2: Attack performance of different methods under different ratios of Byzantine clients. Lower scores represent better attack effectiveness.

Victim Model	Attack Method	MovieLens-1M						Beauty					
		1%		2%		5%		1%		2%		5%	
		HR@5	nDCG@5	HR@5	nDCG@5	HR@5	nDCG@5	HR@5	nDCG@5	HR@5	nDCG@5	HR@5	nDCG@5
NCF	No Attack	0.0135	0.0091	0.0135	0.0091	0.0135	0.0091	0.0125	0.0072	0.0125	0.0072	0.0125	0.0072
	LabelFlip	0.0137	0.0092	0.0133	0.0091	0.0130	0.0090	0.0127	0.0073	0.0122	0.0070	0.0119	0.0069
	Gaussian	0.0138	0.0093	0.0132	0.0091	0.0129	0.0089	0.0125	0.0072	0.0121	0.0070	0.0118	0.0068
	LIE	0.0133	0.0090	0.0127	0.0089	0.0123	0.0087	0.0120	0.0069	0.0118	0.0068	0.0114	0.0065
	STAT-OPT	0.0132	0.0090	0.0125	0.0088	0.0120	0.0086	0.0122	0.0070	0.0119	0.0069	0.0113	0.0065
	DYN-OPT	0.0131	0.0090	0.0126	0.0088	0.0122	0.0087	0.0118	0.0068	0.0115	0.0067	0.0110	0.0063
	FedAttack	0.0124	0.0087	0.0116	0.0083	0.0109	0.0079	0.0112	0.0065	0.0107	0.0064	0.0098	0.0058
BERT4Rec	No Attack	0.0550	0.0291	0.0550	0.0291	0.0550	0.0291	0.0159	0.0161	0.0159	0.0161	0.0159	0.0161
	LabelFlip	0.0544	0.0288	0.0538	0.0285	0.0533	0.0283	0.0155	0.0159	0.0153	0.0157	0.0149	0.0155
	Gaussian	0.0548	0.0289	0.0536	0.0284	0.0532	0.0283	0.0152	0.0156	0.0149	0.0155	0.0144	0.0153
	LIE	0.0534	0.0283	0.0525	0.0279	0.0513	0.0274	0.0149	0.0154	0.0144	0.0152	0.0140	0.0150
	STAT-OPT	0.0536	0.0285	0.0521	0.0277	0.0510	0.0272	0.0150	0.0155	0.0146	0.0153	0.0141	0.0151
	DYN-OPT	0.0530	0.0280	0.0517	0.0272	0.0501	0.0268	0.0146	0.0152	0.0144	0.0151	0.0139	0.0150
	FedAttack	0.0511	0.0270	0.0484	0.0258	0.0459	0.0245	0.0137	0.0149	0.0132	0.0146	0.0122	0.0139

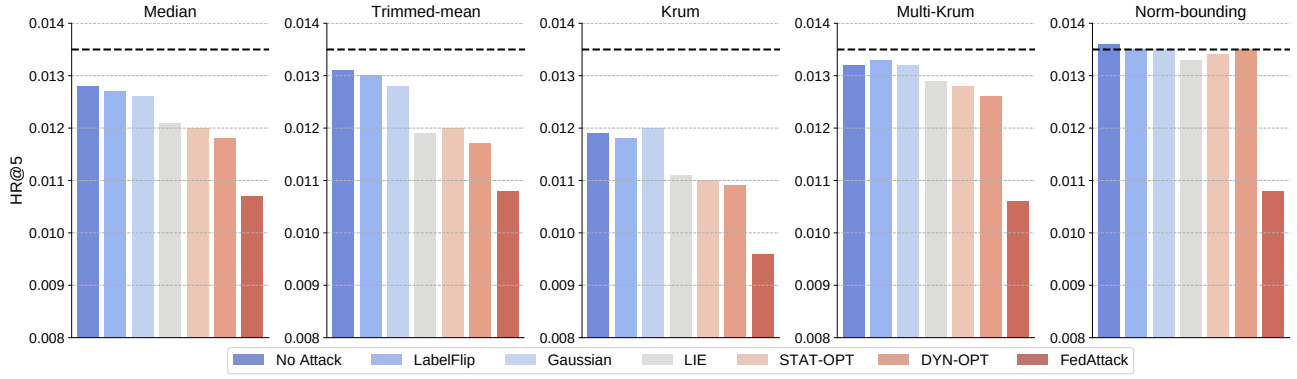


Figure 4: Performance of different attack methods under different defense mechanisms. The black dashed lines represent the performance without attack.

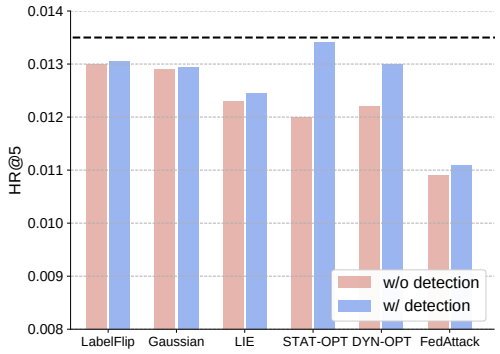
the advantage of our approach in terms of the relative performance degradation can usually be larger when a smaller percentage of Byzantine clients is used. The results show that hard sampling can be a powerful technique for untargeted federated recommendation poisoning with limited malicious clients.

4.3 Attack Effectiveness Under Defense

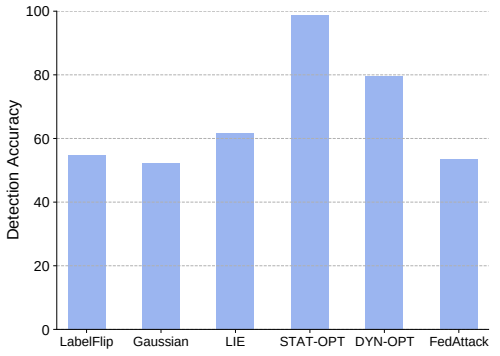
To verify the effectiveness of *FedAttack* in circumventing the defense of robust aggregation mechanisms in FL, we compare the attack performance of different methods when the server uses the following strategies: (1) *Median* [56], using the median value of received gradients for each dimension; (2) *Trimmed-mean* [56], removing some largest and smallest values for each dimension and using the average of the rest; (3) *Krum* [4], selecting the local model that is the most similar to other models as the global model; (4) *Multi-Krum* [4], iteratively selecting several local models through Krum and aggregating them; (5) *Norm-bounding* [40], using a threshold (we use 2 in our experiments) to clip the norm of gradients. We use 5% Byzantine clients in this and the later experiments. The results on the MovieLens-1M dataset are shown in Fig. 4 (the results on Beauty show similar trends and are omitted due to space limit). We find that some defense mechanisms such as *Median*, *Trimmed-mean* and *Krum* have huge impacts on recommendation accuracy, and their performance losses can even be larger than sufferring from some attack methods without defense. Thus, they may not be good choices for defending untargeted poisoning attacks on federated recommendation. *Mutli-Krum* and *Norm-bound* do not substantially hurt the recommendation accuracy, but only *Norm-bounding* can fully defend poisoning attack baselines. This is because *Mutli-Krum* may incorrectly filter some normal gradients, while norm-bounding usually does not hurt the training accuracy if the threshold is appropriate. *FedAttack* cannot be effectively defended by all these robust aggregation methods. This is because it only modifies training samples, and the Byzantine users also have some similar patterns with benign users. Thus, *FedAttack* can circumvent their defense and effectively degrade the model performance.

4.4 Influence of Malicious Gradient Detection

To verify whether *FedAttack* is still effective when there are strong malicious detection mechanisms on the server, we study an extreme



(a) Impacts of malicious gradient detection on attack performance. The black dashed line is the recommendation performance without attack.



(b) Malicious gradient detection accuracy.

Figure 5: Results of malicious gradient detection.

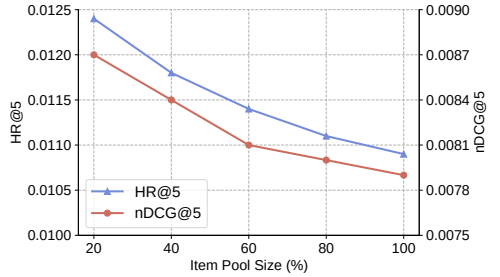


Figure 6: Influence of candidate item pool size.

case where the server detects and filters potential malicious gradients based on strong background knowledge of the attacker.[3] We assume that the server has some samples of malicious gradients for training an anomaly detection model. To simulate this scenario, we first run an experiment to collect malicious gradients. We concatenate the average gradients and the gradients of a client to be classified as the input of a multi-layer feedforward network for detection. Note that we down-sample the normal gradient samples

[3]We do not consider client detection because the client identity may not be exposed to the server in federated learning [21].

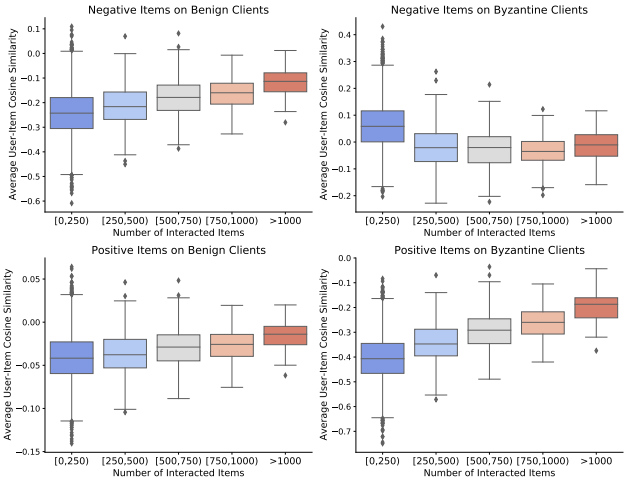


Figure 7: The average similarity between positive/negative items and users w.r.t. different numbers of interacted items on benign and Byzantine clients.

to form a balanced binary classification. After obtaining the detection model, we re-run the federated recommendation learning and use it to detect potential malicious gradients. The server excludes the positively detected gradients before aggregation. The attack performance of different methods on MovieLens-1M and the server's detection accuracy on them are shown in Fig. 5 (the results on Beauty show similar patterns and are omitted). We find that the attack performance of all methods decreases to different extents when there are detection mechanisms, which is intuitive because some malicious clients are filtered. We can see gradient poisoning methods (especially STAT-OPT) are easier to be detected. This is because their gradients usually have similar patterns, and their norm and variance are also larger than normal gradients. In addition, we have an interesting finding that *LabelFlip* and *Gaussian* attacks are difficult to detect, which is different from the observations in general FL [41]. This may be because the user interaction data is quite noisy, and it is hard to distinguish between the patterns of these perturbations and noise. Besides, our *FedAttack* approach can achieve the best attack performance under strong supervised detection mechanisms. It can hardly be detected by the server even the server has prior knowledge of the malicious gradients. Thus, *FedAttack* is even more difficult to be perceived by the server in real-world federated recommendation model training.

4.5 Influence of Item Pool Size

In practice the attacker may not know the entire item set, and the candidate item pool is usually a subset. To study the influence of item pool size on the attack performance, we compare the results of *FedAttack* under different sizes of the item pool. We use randomly sampled subsets of the whole item set to simulate the scenarios when the attacker has partial knowledge of the full item set. The results on MovieLens-1M are shown in Fig. 6 (similar patterns are observed on Beauty). We find that the poisoning effectiveness of *FedAttack* is still satisfactory when a relatively large item subset

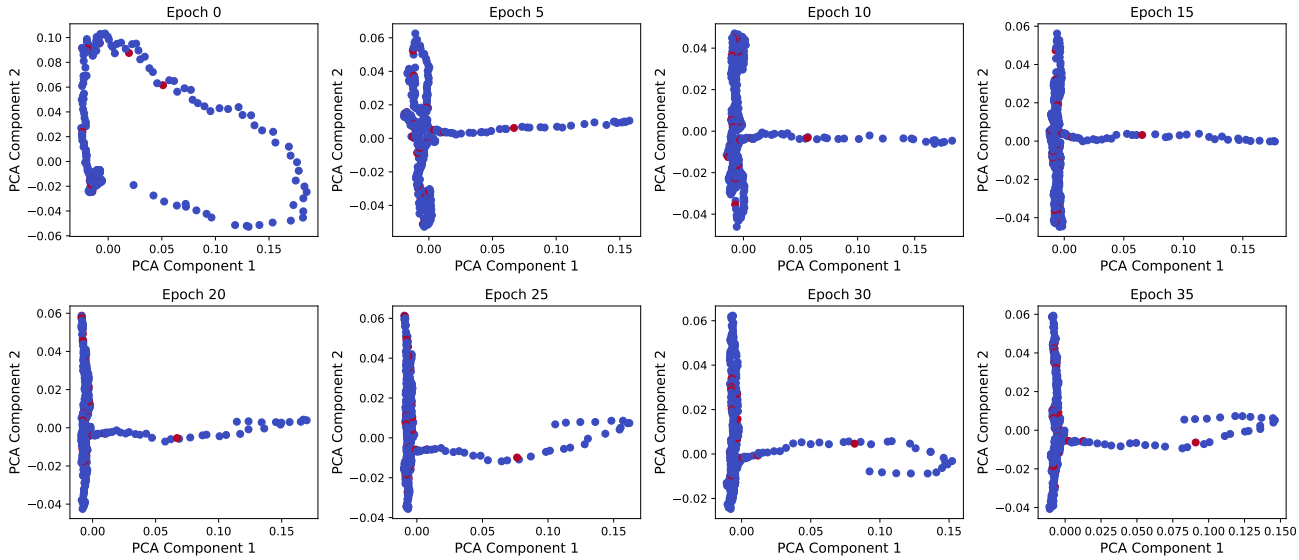


Figure 8: Visualization of local gradients in different epochs of model training. Red and blue points represent gradients from Byzantine and benign clients, respectively.

is used (e.g., 60%), but it is suboptimal when the item pool is too small. This is because the hardest samples in a small subset may not be sufficiently effective to subvert the model training. Thus, the attacker needs to crawl a relatively large item pool to ensure that the hard samples are representative enough.

4.6 Sample Hardness Analysis

To explore whether the hardness of samples on Byzantine clients has significant differences with benign clients, we compare the average similarity between the embeddings of positive or negative items and the embeddings of users with different numbers of interacted items, as shown in Fig. 7. For users with fewer interacted items, both positive and negative samples on benign clients have low average similarities to user embedding, while the similarity variance is large. This is because the interest inferred from the profiles of less active users can be too inaccurate or not comprehensive to predict future behaviors. Different from benign clients, most negative samples on Byzantine clients have higher similarities to user interest while most positive samples are irrelevant to user interest. In addition, we can see that the hardness distribution of positive samples on Byzantine clients with many interacted items has many overlaps with benign but inactive clients. It indicates the behaviors of Byzantine clients have similar patterns with some benign cold users whose behaviors are diverse and noisy. Since the server does not know the raw behavior data distribution on local clients due to privacy reasons, it cannot effectively distinguish between Byzantine and benign clients merely based on their gradients (also see the next section). Thus, *FedAttack* can subvert the model training in a concealed way.

4.7 Gradient Visualization

Finally, we use PCA to visualize the gradients on benign clients and Byzantine clients to see whether they are visually distinguishable, as shown in Fig. 8. From all subplots, we find that malicious gradients have similar PCA component distributions with normal gradients. Thus, it is very difficult to distinguish between normal and malicious gradients. In addition, we find a very interesting phenomenon that there are many outlier gradients during model training and most of them are normal ones. This phenomenon reflects many outlier gradients from benign clients may be mistakenly filtered by defense or detection mechanism, which may heavily impair the recommendation performance on these users. These visualization results further verify the covertness of *FedAttack* and the difficulty in defending it poisoning.

5 CONCLUSION

In this paper, we present an effective untargeted attack method for poisoning federated recommender systems, named *FedAttack*. We propose to employ hard negative sampling as to poison federated recommendation by selecting positive and negative candidate items on Byzantine clients that can subvert the model training. More specifically, we select items that are most similar to the user interest inferred from local user profiles as negative samples, while regarding items that are most dissimilar to user interest as positive samples. The Byzantine clients train the recommendation model on these poisoned samples, and generate malicious gradients to upload to the server for aggregation. Extensive experiments on two benchmark datasets for recommendation validate *FedAttack* can degrade recommendation performance more effectively than many baseline methods. In addition, it can effectively circumvent the existing defense and detection mechanisms for federated learning. Our work shows existing federated recommendation methods can be vulnerable, and the design of future federated recommender systems should be more secure and robust to these attacks, which is one of our future work.

REFERENCES

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *AISTATS*. 2938–2948.

[2] Gilad Baruch, Moran Baruch, and Yoav Goldberg. 2019. A little is enough: Circumventing defenses for distributed learning. In *NeurIPS*, Vol. 32.

[3] Preeti Bhargava, Thomas Phan, Jiayu Zhou, and Juhan Lee. 2015. Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In *WWW*. 130–140.

[4] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NIPS*.

[5] Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. 2019. Understanding distributed poisoning attack in federated learning. In *ICPADS*. IEEE, 233–239.

[6] Chen Chen, Jingfeng Zhang, Anthony KH Tung, Mohan Kankanhalli, and Gang Chen. 2020. Robust federated recommendation system. *arXiv preprint arXiv:2006.08259* (2020).

[7] Huiyuan Chen and Jing Li. 2019. Data poisoning attacks on cross-domain recommendation. In *CIKM*. 2177–2180.

[8] Jian Chen, Xuxin Zhang, Rui Zhang, Chen Wang, and Ling Liu. 2021. De-pois: An attack-agnostic defense against data poisoning attacks. *TIFS* 16 (2021), 3412–3425.

[9] Clarence Chio and David Freeman. 2018. *Machine learning and security: Protecting systems with data and algorithms.* " O'Reilly Media, Inc.".

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

[11] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. 2018. Poisoning attacks to graph-based recommender systems. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 381–392.

[12] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *TIIS* 5, 4 (2015), 1–19.

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.

[15] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. 2020. Hard negative mixing for contrastive learning. In *NeurIPS*, Vol. 33. 21798–21809.

[16] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE, 197–206.

[17] Walid Krichene and Steffen Rendle. 2020. On sampled metrics for item recommendation. In *KDD*. 1748–1757.

[18] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. *NIPS* 29 (2016).

[19] Feng Liang, Weike Pan, and Zhong Ming. 2021. Fedrec++: Lossless federated recommendation with explicit feedback. In *AAAI*, Vol. 35. 4224–4231.

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

[21] Ruixuan Liu, Yang Cao, Hong Chen, Ruoyang Guo, and Masatoshi Yoshikawa. 2021. FLAME: Differentially Private Federated Learning in the Shuffle Model. In *AAAI*, Vol. 35. 8688–8696.

[22] Lingjuan Lyu, Han Yu, Xingjun Ma, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S Yu. 2020. Privacy and robustness in federated learning: Attacks and defenses. *arXiv preprint arXiv:2012.06337* (2020).

[23] Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133* (2020).

[24] Atulay Mahajan and Sangeeta Sharma. 2015. The malicious insiders threat in the cloud. *IJERGS* 3, 2 (2015), 245–256.

[25] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.

[26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*. 1273–1282.

[27] Luis Muñoz-González, Bjarne Pfitzner, Matteo Russo, Javier Carnerero-Cano, and Emil C Lupu. 2019. Poisoning attacks with generative adversarial nets. *arXiv preprint arXiv:1906.07773* (2019).

[28] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N Asokan, and Ahmad-Reza Sadeghi. 2019. DÏoT: A federated self-learning anomaly detection system for IoT. In *ICDCS*. IEEE, 756–767.

[29] Ivan Palomares, Fiona Browne, and Peadar Davis. 2018. Multi-view fuzzy information fusion in collaborative filtering recommender systems: Application to the urban resilience domain. *Data & Knowledge Engineering* 113 (2018), 64–80.

[30] Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2020. Privacy-Preserving News Recommendation Model Learning. In *EMNLP: Findings*. 1423–1432.

[31] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. 2021. Adaptive federated optimization. In *ICLR*.

[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.

[33] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2020. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592* (2020).

[34] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. 2021. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *ICML*. 9389–9398.

[35] Virat Shejwalkar and Amir Houmansadr. 2021. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*.

[36] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. 2022. Back to the drawing board: A critical evaluation of poisoning attacks on federated learning. In *S&P*.

[37] Junshuai Song, Zhao Li, Zehong Hu, Yucheng Wu, Zhenpeng Li, Jian Li, and Jun Gao. 2020. Poisonrec: an adaptive data poisoning framework for attacking black-box recommender systems. In *ICDE*. IEEE, 157–168.

[38] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. 2017. Certified defenses for data poisoning attacks. *NIPS* 30 (2017).

[39] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*. 1441–1450.

[40] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. 2019. Can you really backdoor federated learning?. In *NeurIPS FL Workshop*.

[41] Rahim Taheri, Reza Javidan, Mohammad Shojafar, Zahra Pooranian, Ali Miri, and Mauro Conti. 2020. On defending against label flipping attacks on malware detection systems. *Neural Computing and Applications* 32, 18 (2020), 14781–14800.

[42] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting adversarially learned injection attacks against recommender systems. In *Recsys*. 318–327.

[43] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *ESORICS*. 480–501.

[44] Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. 2021. Concealed Data Poisoning Attacks on NLP Models. In *NAACL*.

[45] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising implicit feedback for recommendation. In *WSDM*. 373–381.

[46] Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, and Enhong Chen. 2021. Triple Adversarial Learning for Influence based Poisoning Attack in Recommender Systems. In *KDD*. 1830–1840.

[47] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019. Npa: Neural news recommendation with personalized attention. In *KDD*. 2576–2584.

[48] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925* (2021).

[49] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Is News Recommendation a Sequential Recommendation Task? *arXiv:2108.08984* (2021).

[50] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. 2015. Support vector machines under adversarial label contamination. *Neurocomputing* 160 (2015), 53–62.

[51] Han Xiao, Huang Xiao, and Claudia Eckert. 2012. Adversarial label flips attack on support vector machines. In *ECAI*. 870–875.

[52] Hong Xuan, Abby Stylianou, Xiaotong Liu, and Robert Pless. 2020. Hard negative examples are hard, but useful. In *ECCV*. Springer, 126–142.

[53] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. 2017. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340* (2017).

[54] Liu Yang, Ben Tan, Vincent W Zheng, Kai Chen, and Qiang Yang. 2020. Federated recommendation systems. In *Federated Learning*. Springer, 225–239.

[55] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *TIST* 10, 2 (2019), 1–19.

[56] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*. 5650–5659.

[57] Wenhui Yu and Zheng Qin. 2020. Sampler design for implicit feedback data by noisy-label robust learning. In *SIGIR*. 861–870.

[58] Hengtong Zhang, Yaliang Li, Bolin Ding, and Jing Gao. 2020. Practical data poisoning attack against next-item recommendation. In *WWW*. 2458–2464.

[59] Hengtong Zhang, Changxin Tian, Yaliang Li, Lu Su, Nan Yang, Wayne Xin Zhao, and Jing Gao. 2021. Data Poisoning Attack against Recommender System Using Incomplete and Perturbed Data. In *KDD*. 2154–2164.

[60] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Quoc Viet Hung Nguyen, and Lizhen Cui. 2021. PipAttack: Poisoning Federated Recommender Systems forManipulating Item Promotion. *arXiv preprint arXiv:2110.10926* (2021).

[61] Yihe Zhang, Xu Yuan, Jin Li, Jiadong Lou, Li Chen, and Nian-Feng Tzeng. 2021. Reverse Attack: Black-box Attacks on Collaborative Recommendation. In *SIGSAC*. 51–68.