Single-User Injection for Invisible Shilling Attack against Recommender Systems

Chengzhi Huang
Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China
Xiamen University
Xiamen, China
edisonchen@stu.xmu.edu.cn

Hui Li*
Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China
Xiamen University
Xiamen, China
hui@xmu.edu.cn

ABSTRACT

Recommendation systems (RS) are crucial for alleviating the information overload problem. Due to its pivotal role in guiding users to make decisions, unscrupulous parties are lured to launch attacks against RS to affect the decisions of normal users and gain illegal profits. Among various types of attacks, shilling attack is one of the most subsistent and profitable attacks. In shilling attack, an adversarial party injects a number of well-designed fake user profiles into the system to mislead RS so that the attack goal can be achieved. Although existing shilling attack methods have achieved promising results, they all adopt the attack paradigm of multi-user injection, where some fake user profiles are required. This paper provides the first study of shilling attack in an extremely limited scenario: only one fake user profile is injected into the victim RS to launch shilling attacks (i.e., single-user injection). We propose a novel single-user injection method SUI-Attack for invisible shilling attack. SUI-Attack is a graph based attack method that models shilling attack as a node generation task over the user-item bipartite graph of the victim RS, and it constructs the fake user profile by generating user features and edges that link the fake user to items. Extensive experiments demonstrate that SUI-Attack can achieve promising attack results in single-user injection. In addition to its attack power, SUI-Attack increases the stealthiness of shilling attack and reduces the risk of being detected. We provide our implementation at: https://github.com/KDEGroup/SUI-Attack.

CCS CONCEPTS

• **Security and privacy → Web application security**; • **Information systems → Recommender systems**.

KEYWORDS

Shilling Attack, Recommender System, Adversarial Attack

*Corresponding author.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3615062

ACM Reference Format:
Chengzhi Huang and Hui Li. 2023. Single-User Injection for Invisible Shilling Attack against Recommender Systems. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3615062

1 INTRODUCTION

With the rapid development of information technology, we are facing a huge growth of available information, causing the information overload problem [3]: it is difficult to effectively make decisions when facing too much information. Recommender systems (RS) are an essential tool to alleviate information overload and have been widely deployed in e-commerce platforms (e.g., Amazon and Taobao) and content-providing platforms (e.g., TikTok and YouTube), bringing massive revenue [59].

However, the prevalence of RS has also attracted unscrupulous parties [11]. They try to attack RS to can gain illegal profits. Among various attack types, shilling attack is one of the most subsistent and profitable attacks against RS [50]. In shilling attack, an adversarial party injects a number of well-designed fake user profiles into the system to mislead RS so that the attack goal can be achieved [11, 18, 47]. One main attack goal is to promote a target item: increase the possibility that the target item can be viewed/bought by people. Studying how to spoof RS has become a hot direction in the RS community as it gives insights into improving the defense against malicious attacks [64].

Much effort has been devoted to designing shilling attack methods. Pioneering works (e.g., Random Attack [5], Bandwagon Attack [4] and Segment Attack [5]) mainly adopt heuristics [18]. Recently, based on the idea of adversarial attack [62], a great number of shilling attack approaches have sprung up, including but not limited to optimization based methods [50], GAN based methods [36, 37], reinforcement learning based methods [48], knowledge distillation based methods [63] and pre-training based methods [64]. Existing methods all adopt the same attack paradigm: inject some fake user profiles into the victim RS. We name such an attack paradigm *multi-user injection*. As more injected fake user profiles typically improve the attack performance but increase the risk of being detected, the number of the injected fake user profiles is typically not large, e.g., 50.

Although existing shilling attack methods have achieved promising attack performance [11], they all assume there is a trade-off between the number of fake users and the performance of the attack. To our best knowledge, no work has studied and answered a

critical question about shilling attack: How many fake user profiles are required to launch a successful shilling attack? In this paper, we study shilling attack in an extremely limited scenario: only one fake user profile is injected into the victim RS to launch shilling attacks (i.e., *single-user injection*). Adversarial attacks against different AI models in the extremely restricted settings (e.g., one-pixel attack in image classification [49] and single-node attack against Graph Neural Networks [16, 51]) have attracted considerable attention since they unveil the severe vulnerability of AI models: adversary can hoax the model with minimum effort. Single-user attacks against RS, if possible, will result in the virtually undetectable attack as it is extremely difficult to identify the only fake user from plenty of real users.

In this paper, we propose a novel single-user injection method for invisible shilling attack against RS (i.e., SUI-Attack). SUI-Attack is a graph based attack method that models shilling attack as a node generation task over the user-item bipartite graph of the victim RS. SUI-Attack contains two phases: feature generation and edge generation. The feature generation phase aims to produce toxic fake user features that can guide the generation of edges that are connected to the fake user. The edge generation phase connects the fake user to items in the user-item bipartite graph to ensure the injected fake user can affect the victim RS, and it is equivalent to filling the fake user profile with interaction history in contemporary shilling attack approaches. The contributions of this work can be summarized as follows:

- We propose the idea of single-user injection. As far as we know, this is the first work to study shilling attacks in the extremely restricted scenario.
- We design a novel single-user injection method SUI-Attack, increasing the stealthiness of shilling attacks and reducing the risk of being detected. SUI-Attack models the single-user injection task over the user-item bipartite graph and constructs the fake user profile by generating its user features and edges that link the fake user to items.
- We conduct extensive experiments to demonstrate that SUI-Attack can achieve promising attack results in single-user injection. In other words, shilling attacks against RS with single-user injection is achievable. Furthermore, in the traditional multi-user injection setting, SUI-Attack is shown to be effective and can cause comparable attack results compared to existing shilling attack methods, showing its flexibility in shilling attacks.

The remaining parts of this paper are organized as follows: Section 2 provides the background knowledge of this work. We describes the details of SUI-Attack in Section 3. Experimental results and analysis are presented in Section 4. Section 5 introduces the related work of this study. Section 6 concludes this work.

2 BACKGROUND

In this section, we first provide some background knowledge of shilling attack and the extremely restricted setting of shilling attack (i.e., single-user injection) that we consider in this paper.

Attack Goal: There are two types of shilling attack: promotion attack and nuke attack [18, 47]. Through injecting several fake user profiles into the target RS (i.e., the *victim RS*), promotion attack aims to improve the ranking of the *target item* in a user's recommendation list. The goal of nuke attack is opposite to promotion attack and it can be easily achieved by reversing the goal of the promotion attack. Hence, for simplicity, we focus on the promotion attack in this paper.

After a successful shilling attack, the target item should appear in as many users' recommendation lists as possible while the overall recommendation performance of the system is not affected [28]. In addition to the traditional settings of shilling attack, in this paper, we impose an extremely restricted constraint on shilling attack and study the single-user-injection shilling attack: the attacker only inject one fake user profile to spoof RS.

Attack Knowledge: We consider the most common setting of attack knowledge used by the existing studies of shilling attack [9, 36]. The attackers do not have prior knowledge of the model architecture of the victim RS. They cannot access the parameters of the victim RS model as well as the gradients during training. However, attackers can access the most basic user-item historical data of the victim RS, i.e., user-item ratings. User-item ratings in many RS (e.g., Amazon) are publicly accessible and can be crawled by attackers.

Attack Capabilities: Typically, a more powerful shilling attack requires injecting more fake user profiles, making the attack more perceptible to the system. Therefore, the number of fake user profiles and the number of maximum interacted items in each fake user profile are limited to b_{user} and b_{item} (i.e., the budget), respectively. Note that, in the setting of single-user injection studied in this paper, b_{user} is set to 1. However, we still report the case when $b_{user} > 1$ so that SUI-Attack can be compared to other shilling attack methods in the traditional multi-user injection setting.

3 OUR METHOD SUI-ATTACK

In this section, we illustrate the details of our proposed SUI-Attack.

3.1 Overview

We model the single-user-injection attack as a node generation process over the user-item bipartite graph. The target is to generate a fake user node that can be used to guide the construction of the fake user profile for injection. The user-item bipartite graph, where each edge between a user node and an item node indicates an historical user-item interaction and edge weights denote interaction features like ratings, is commonly used to model RS.

SUI-Attack uses two phases, feature generation and edge generation, to generate the fake user profile, including its user features and edges connecting the fake user and items on the bipartite graph, for single-user-injection attack.

3.2 Feature Generation

User features (e.g., statistics of historical ratings), which are typically used to initialize the embedding layer in RS models, are an important source for RS to model user preferences. Attackers can also leverage user features to guide the construction of the fake user profile. However, unlike real users, the fake user does not have features as it does not have real interaction history. Feature generation phase aims to generate fake user's node features with strong toxicity that can guide the subsequent edge generation phase to

generate destructive user-item interactions for the fake user to hoax RS. Note that *user features in fake user profile construction are not the same as user features modeled by the victim RS.* The latter are unacquirable for attackers as they cannot access the details of the victim RS model.

3.2.1 Selection of Node Features. As different RS may have their own designed user/item features, we choose to adopt 10 prevalent RS features [41, 56] that rely on the intrinsic information of the user-item bipartite graph and do not require specific user or item information (e.g., user demographics and item descriptions). This way, SUI-Attack is not limited to specific feature designs and can be applied to different RS. The definitions of the ten chosen features are as follows:

(1) **Rating Deviation from Mean Agreement (RDMA)** measures the average deviation of a user's ratings from the mean agreement for a set of target items:

$$\text{RDMA}_u = \frac{\sum_{i \in N(u)} \frac{|r_{u,i} - \bar{r}_i|}{M_i}}{|N_u|}, \tag{1}$$

where N_u is the items that user u has rated, $|N_u|$ is the number of items in N_u (i.e., profile size), M_i is the number of ratings received by the item i, $r_{u,i}$ denotes the ratings given by user u on item i, and $\bar{r}_i$ denotes the mean rating of item i. The reciprocal of the number of ratings for each item (M_i) is used as a weight since items with more ratings are more likely to be rated accurately and the weights of their deviations should be reduced.

(2) **Length Variance (LengthVar):** LengthVar measures the variance of the number of interactions in a user's profile (i.e., profile size) and it is defined as follows:

$$\text{LengthVar}_u = \frac{|N_u| - |\bar{N}|}{\sum_{j \in U} (|N_j| - |N_u|)^2}, \tag{2}$$

where U indicates the user set in RS and $|\bar{N}|$ is the average profile size in RS.

(3) **Filler Mean Variance (FMV):** FMV measures the deviation of a user's rating in a hypothesized filler partition from the mean rating for each item. The hypothesized filler partition contains randomly sampled items. We sample at most 50 items for each user profile as the hypothesized filler partition. Then, FMV is defined as:

$$\text{FMV}_u = \frac{1}{|H_u|} \sum_{i \in H_u} (r_{u,i} - \bar{r}_i)^2, \tag{3}$$

where H_u is the hypothesized filler partition in the user profile of u and $|H_u|$ indicates the number of items in H_u.

(4) **Filler Average Correlation (FAC)** measures the correlation between the rating of an item in the hypothesized filler partition of a user profile and the item's average rating:

$$\text{FAC}_u = \frac{\sum_{i \in H_u} (r_{u,i} - \bar{r}_i)}{\sqrt{\sum_{j \in H_u} (r_{u,j} - \bar{r}_j)^2}}. \tag{4}$$

(5) **Mean Variance (MeanVar)** calculates the average variance between the items in the hypothesized filler partition and their average ratings:

$$\text{MeanVar}_u = \frac{\sum_{i \in F_u} (r_{u,i} - \bar{r}_u)^2}{|F_u|}, \tag{5}$$

where F_u contains all the items that user u did not give the maximal rating score $r_{\max}$. For example, $r_{\max}$ is 5 in a five-scale rating system. $\bar{r}_u$ indicates the average rating of all ratings given by u.

(6) **Filler Mean Target Difference (FMTD)** quantifies the discrepancy between the maximal rating score $r_{\max}$ and rating scores provided by user u that are not maximal:

$$\text{FMTD}_u = \left| \frac{\sum_{i \in M_u} r_{\max}}{|M_u|} - \frac{\sum_{k \in F_u} r_{u,k}}{|F_u|} \right|, \tag{6}$$

where M_u indicates the items that u gave the maximal rating score $r_{\max}$.

(7) **Filler Size with Total Items (FSTI)** is the percentage of a user' profile size over the number of items in the RS:

$$\text{FSTI}_u = \frac{|N_u|}{|I|}, \tag{7}$$

where I is the item set in RS.

(8) **Filler Size with Popular Items in Itself (FSPII)** is the percentage of most popular items that a user has rated over the profile size:

$$\text{FSPII}_u = \frac{\sum_{i \in I_p} \mathbb{I}_1(u, i)}{|N_u|} \tag{8}$$

where V_p is the most popular items in RS and we define it as the top 5% most popular items (with many interactions) in RS. $\mathbb{I}_1(u, i)$ is 1 if user u has rated item i; otherwise 0.

(9) **Filler Size with Maximum Rating in Itself (FSMAXRI)** indicates the percentage of the times that a user u gave the maximal rating score over u's profile size:

$$\text{FSMAXRI}_u = \frac{\sum_{i \in N_u} \mathbb{I}_2(r_{u,i}, r_{\max})}{|N_u|}, \tag{9}$$

where the indicator $\mathbb{I}_2(r_{u,i}, r_{\max})$ is 1 if $r_{u,i}$ equals $r_{\max}$; otherwise 0.

(10) **Filler Size with Average Rating in Itself (FSARI)** indicates the percentage of the times that a user u gave the average rating score over u's profile size:

$$\text{FSARI}_u = \frac{\sum_{i \in I} \mathbb{I}_3(r_{u,i}, r_{\text{avg}})}{|N_u|} \tag{10}$$

where r_{avg} is the global average score in RS. The indicator $\mathbb{I}_3(r_{u,i}, r_{\text{avg}})$ is 1 if the floor or the ceiling of r_{avg} equals $r_{u,i}$; otherwise 0.

Note that, although we illustrate the definitions of the selected features from user side, they can be used as both user features and item features. Based on the selected features, for each user/item, we construct a normalized 10-dimensional feature vector $\mathbf{x}$.

3.2.2 Generate Toxic Fake User Features. Given features of real users and items, the next step is to generate toxic fake user features that can guide the edge generation to fill the fake user profile with user-item interaction data. To this end, SUI-Attack adopts the idea of reconstruction: train a graph encoder by predicting the features of

some masked real users and items, and then use the graph encoder to predict the features of the fake user.

Specifically, SUI-Attack first maps features of real users and items into high dimensional representation spaces via a two-layer feedforward neural network:

$$\mathbf{p} = \mathbf{W}_2 \cdot \text{LeakyRELU}(\mathbf{W}_1\mathbf{x}), \tag{11}$$

where $\mathbf{x}$ is the feature vector of a user or an item, $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable parameters.

Then, SUI-Attack uses a multi-relation graph convolution layer to aggregate information of neighboring nodes and update representations for each user in the user-item bipartite graph:

$$\begin{aligned} \mathbf{q}_u &= \sum_{r=1}^{2} \sum_{v \in \mathcal{N}_r(u)} \frac{\mathbf{W}_r \mathbf{p}_v}{\sqrt{|\mathcal{N}_r(u)| \cdot |\mathcal{N}_r(v)|}} \\ \mathbf{h}_u &= \text{LeakyRELU}\big(\mathbf{W}_1 \cdot \text{LeakyRELU}(\mathbf{q}_u)\big) \end{aligned} \tag{12}$$

where $\mathcal{N}_r(u)$ indicates the neighboring node of u w.r.t. to edge type r and there are two types of edges (user→item edges and item→user edges). SUI-Attack uses a similar aggregation process for updating item representations.

Next, SUI-Attack randomly masks some real user and item nodes and reconstructs the masked features. SUI-Attack uses a two-layer MLP for the feature reconstruction:

$$\hat{\mathbf{x}} = \mathbf{W}_2^{(rec)} \cdot \sigma\big(\mathbf{W}_1^{(rec)}\mathbf{h}\big), \tag{13}$$

where $\mathbf{W}_1^{(rec)}$ and $\mathbf{W}_2^{(rec)}$ are trainable parameters. Suppose that the masked user set is U_m and the masked item node set is V_m, the following reconstruction loss is used for feature reconstruction:

$$\mathcal{L}_{\text{recon}} = \frac{1}{|U_m|} \sum_{u \in U_m} ||\mathbf{x}_u - \hat{\mathbf{x}}_u||^2 + \frac{1}{|V_m|} \sum_{v \in V_m} ||\mathbf{x}_v - \hat{\mathbf{x}}_v||^2, \tag{14}$$

where $\mathbf{x}_u$ is the features of user $u \in U_m$ and $\hat{\mathbf{x}}_u$ is the reconstructed features of u. Similar notations $\mathbf{x}_v$ and $\hat{\mathbf{x}}_v$ are used for reconstructing item features.

Through reconstruction, the graph encoder is empowered by the capability to encode topological and feature information of the bipartite graph containing real users and items, and it can be used to generate features for the fake user. We initialize the feature vector $\mathbf{x}_{z'}$ of the fake user z' as zero vector and use SUI-Attack to predict $\hat{\mathbf{x}}_{z'}$ (Equation 13) as fake user features. However, the generated features for the fake user do not convey toxicity and cannot guide the edge generation to fulfill the attack goal. Therefore, we further adopt the idea of influence functions [25], a classic technique from robust statistics [10] that has shown promising results in determining the importance of a training sample in RS [58], to endow the generated features of the fake user with destructive power.

To be specific, influence functions show how the model parameters change as we upweight a training sample by an infinitesimal amount. For a training sample z, if it is upweighted by a small value ϵ, the changed parameter $\hat{\theta}_{\epsilon,z}$ can be defined as:

$$\hat{\theta}_{\epsilon,z} \overset{\text{def}}{=} \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\text{RS}}(z_i, \theta) + \epsilon \mathcal{L}_{\text{RS}}(z, \theta), \tag{15}$$

where $\mathcal{L}(\cdot)_{\text{RS}}$ indicates the training loss of RS and n is the number of samples. Then, the influence of upwerighting z on the parameter is given by:

$$\mathcal{I}_{\text{up,params}}(z) \overset{\text{def}}{=} \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}_{\text{RS}}(z, \hat{\theta}), \tag{16}$$

where $H_{\hat{\theta}}^{-1} = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \mathcal{L}_{\text{RS}}(z, \hat{\theta})$ is the Hessian matrix of $\mathcal{L}_{\text{RS}}$. Based on Eq. 16, Koh and Liang [25] derivate that the influence of upweighting z on a test sample z_{test} has a closed-form expression:

$$\mathcal{I}_{\text{up,loss}}(z, z_{test}) = -\nabla_\theta \mathcal{L}(z_{test}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}_{\text{RS}}(z, \hat{\theta}). \tag{17}$$

We use Equation 17 to pre-compute the influence scores of all real users. We then use a three-layer feedforward neural network as an influence predictor $\text{IP}(\cdot)$. Given the feature vector x_z of a real user z, the influence predictor is trained to predict the influence score of z by minimizing the gap (mean squared error) between the true influence score and the predicted influence score.

In summary, the optimization objective for generating toxic fake user features can be formulated as:

$$\mathcal{L}_{\text{feat}} = \mathcal{L}_{\text{recon}} - \text{IP}(x_{z'}). \tag{18}$$

Minimizing the loss function in Equation 18 trains SUI-Attack to reconstruct node features more accurately and maximizing the influence of the fake user z' at the same time.

3.3 Edge Generation

An essential step in contemporary shilling attack approaches is filling the fake user profile with interaction history. This step is equivalent to connecting the fake user node with items in the user-item bipartite graph which ensures the fake user profile can affect the recommendation of the RS on the target item.

Specifically, we use the generated features of the fake user profile to guide the generation of edges. We first project the predicted fake user features (Equation 11) and the features of candidate items (i.e., the ten selected RS features) through a single layer feedforward neural network in order to project them to the same space:

$$\mathbf{q}_{z'} = \mathbf{W}_{\text{edge}}\hat{\mathbf{x}}_{z'}, \ \mathbf{q}_j = \mathbf{W}_{\text{edge}}\mathbf{x}_j \tag{19}$$

where $\mathbf{W}_{\text{edge}}$ is trainable parameters and j is a candidate item. We choose the 2-hop item neighbors of the target item as the candidate items. Candidate items and the target item were interacted by same real users in the past. According to the idea of co-visitation attack [61], these candidate items can affect whether the target item can be recommended after shilling attack. To avoid being easily detected, we additionally add s sampled popular items into the candidate set.

Then, we calculate the probability of connecting the fake user to each candidate item by measuring the the cosine similarity between $\mathbf{q}_{z'}$ and $\mathbf{q}_j$. The resulting probability distribution $\mathbf{o} \in \mathbb{R}^{b_{\text{item}}}$ contains probabilities of all the candidate items. Next, our target is to choose top-b_{item} candidate items with the highest probabilities. To address the discretization issue of the network, we employ the Gumbel-Top-K technique. It is an extension of the Gumbel-Max trick for sampling from a categorical distribution. The Gumbel-Max trick is a method that adds independent and identically distributed (i.i.d.) Gumbel noise to the log-probabilities of each category and selects the category with the highest sum of log-probability and Gumbel noise [13, 22]. The Gumbel-Top-k trick extends this method to

sample k elements without replacement. For $\varepsilon \sim \text{Uniform}(0,1)$, Gumbel-Softmax is defined as:

$$\text{Gumble-Softmax}(\mathbf{o})_i = \frac{\exp(\frac{(\log(o_i)+g_i)}{\tau})}{\sum_{j=1}^{c}\exp(\frac{(\log(o_j)+g_j)}{\tau})}, \tag{20}$$

where m is the size of the candidate item set. where the parameter $\tau > 0$ represents the annealing factor that determines how close the output result is to the one-hot form. A smaller τ value leads to a more one-hot-like output, but may cause a more severe gradient vanishing problem. o_i is the i-th dimension in $\mathbf{o}$. The Gumbel distribution $g_i = -\log(-\log \varepsilon_i)$ and it brings exploration to the edge selection process. And we can further use α to control the strength of exploration:

$$\text{Gumble-Softmax}(\mathbf{o}, \epsilon)_i = \frac{\exp(\frac{(\log(o_i)+\alpha\cdot g_i)}{\tau})}{\sum_{j=1}^{n}\exp(\frac{(\log(o_j)+\alpha\cdot g_j)}{\tau})}. \tag{21}$$

The Gumbel-Top-K function for edge generation can be formulated as follow:

$$G(\mathbf{o}) = \sum_{i=1}^{b_{\text{item}}} \text{Gumble-Softmax}(\mathbf{o} \odot \text{mask}_i, \alpha)_i, \tag{22}$$

where mask_i filters out the selected edges so that they are not chosen again in subsequent iterations. Note that the resulting vector is sharp but not strictly discrete, which facilitates the training process [51]. In the test phase, we enforce a hard threshold e on the vector to choose edges that connect to the fake user.

3.4 Optimization

We inject the generated fake user node into the user-item bipartite graph to launch shilling attacks. We design the following attack loss to endow the generated fake user with strong destructive power:

$$\mathcal{L}_{adv}(X, \hat{\theta}) = -\sum_{u \in \mathcal{U}} log(\frac{exp(r_{u,t})}{\sum_{j \in \mathcal{I}} exp(r_{u,j})}), \tag{23}$$

where t indicates the target item. Equation 23 shows the attack goal of promotion shilling attack: hoax RS and mislead RS to rank the target item higher than other items when making recommendation.

In summary, the complete objective of SUI-Attack is:

$$\mathcal{L} = \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{feat}} = \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{recon}} - \text{IP}(x_{z'}). \tag{24}$$

And SUI-Attack can be optimized using gradient descent based methods like Adam [24]. When the optimization of SUI-Attack finishes, we can construct a fake user in the victim RS and fill the fake user profile with some user-item interactions guided by the generated edges from SUI-Attack. Then, the victim RS will be affected by the injected fake user and the attack goal can be achieved.

4 EXPERIMENT

In this section, we present the experimental results and analysis.

4.1 Experimental Settings

4.1.1 Dataset. To demonstrate the effectiveness of our method, we conducted experiments on three public datasets Automotive[1], Tools & Home Improvement (T & HI)[1] and Last.fm[2] that are widely used in previous studies of shilling attacks and RS [7, 36, 37]. Table 1 provides the statistics of the data. We randomly choose 5 items from each dataset as the target items.

Table 1: Statistics of datasets

Dataset	Users	Items	Interactions	Sparsity
Automotive	2,928	1,835	20,473	99.62%
T & HI	1,208	8,491	28,396	99.72%
Last.fm	1,892	12,523	186,479	99.21%

4.1.2 Baselines. We compare SUI-Attack with traditional shilling attack methods Random Attack [5], Bandwagon Attack [4] and Segment Attack [5], and state-of-the-art deep learning based methods, including TrialAttack [58], AUSH [36] and LegUP [37]. We use the implementation[3] provided by the original authors for TrialAttack. For other methods, we use the implementations[1] provided by Lin et al. [37]. We follow the recommended settings of each method.

4.1.3 Victim RS. We conduct shilling attacks against prevalent RS including traditional recommendation models (ItemCF [43] and WMF [21]) and deep learning based recommendation models (NGCF [55], VAE [35], ItemAE [44], LightGCN [19] and NCF [20]). We refer to their original papers for parameter settings.

4.1.4 Evaluation Metric. We adopt Hit Ratio (HR@k), a metric that is widely used to evaluate the performance of shilling attack [58]. It indicates the fraction of users for whom the top-k recommendation list after the attack contains the target item. We set k to 50 in our experiments.

4.1.5 Parameter Settings. In feature generation, we randomly mask 10% of nodes for recovery, the candidate items are 2-hop neighboring items of the target item and the 5% items (i.e., s) sampled from the top 10% popular items in the RS. In edge generation, we set $e = 0.85$ to choose the edges that connect to the fake user. In addition to testing the performance when injecting only one fake user, we also analyze the results when multiple fake users are injected into the RS so that SUI-Attack can be compared to contemporary shilling attack methods, and we set the number of fake users (b_{user}) to 50 in multi-user injection. Fake user(s) in both single-user injection and multi-user injection connect to 50 (b_{item}) items at most. We set the training epoch to 64 and batch size to 32. The learning rate is set to 5e-4 and we adopt cosine decay learning rate scheduler. We use gradient normalization and the max norm is set to 1.0. For the graph encoder, we apply a dropout layer to the input of a GCN layer with a dropout rate of 0.5. The hidden size of the graph encoder is set to 250. We select LeakyReLU as the non-linear activate function with the negative slope being 0.1.

4.2 Performance of Shilling Attack

Table 2 presents the results of our method and the baseline models, and the best results are shown in bold. The value on the left side of the slash indicates HR@50 of single-user injection using each method. For single-user injection, baselines are modified to inject

[1] https://github.com/XMUDM/ShillingAttack

[2] https://grouplens.org/datasets/hetrec-2011

[3] https://github.com/ustcml/TrialAttack

Table 2: Attack performance (HR@50) of different attack methods against different victim RS. The left side of the slash is the attack performance for single-user injection, and the right side is the attack performance for multi-user injection. Best results of single-user injection and multi-user injection are shown in bold.

Dataset	Victim RS	Shilling Attack Methods										
		Random	Segment	Bandwagon	TrialAttack	AUSH	LegUP	SUI-Attack	ratio 1	ratio 2	ratio 3	ratio 4
T & HI	ItemCF	0.010/0.207	0.000/0.126	0.000/0.122	0.042/**0.295**	0.012/0.172	0.040/0.253	**0.194**/0.262	0.658	4.620	0.740	0.888
	WMF	0.000/0.063	0.000/0.020	0.000/0.024	0.000/**0.081**	0.004/0.046	0.000/0.050	**0.007**/0.062	0.086	1.750	0.113	0.765
	NGCF	0.001/0.093	0.001/0.090	0.001/**0.102**	0.000/0.071	0.002/0.090	0.000/0.101	**0.076**/0.092	0.745	38.00	0.826	0.902
	VAE	0.203/0.762	0.174/0.826	0.000/0.811	0.103/**0.992**	0.241/0.962	0.227/0.931	**0.636**/0.937	0.641	2.639	0.679	0.945
	ItemAE	0.014/0.234	0.001/0.143	0.000/0.174	0.004/**0.281**	0.082/0.145	0.002/0.268	**0.168**/0.192	0.598	2.049	0.875	0.683
	LightGCN	0.000/0.027	0.000/0.048	0.000/**0.151**	0.000/0.039	0.000/0.053	0.000/0.037	**0.023**/0.034	0.433	$+\infty$	0.676	0.225
	NCF	0.301/**0.883**	0.193/0.717	0.163/0.783	0.265/0.690	0.317/0.804	0.187/0.824	**0.481**/0.862	0.545	1.517	0.545	0.976
Last.FM	ItemCF	0.000/0.201	0.002/0.112	0.000/0.109	0.003/0.211	0.003/0.191	0.012/**0.227**	**0.117**/0.208	0.555	9.750	0.563	0.916
	WMF	0.001/0.104	0.010/0.090	0.010/0.163	0.095/**0.201**	0.020/0.182	0.012/0.155	**0.118**/0.192	0.587	1.242	0.615	0.955
	NGCF	0.084/0.382	0.067/0.287	0.051/0.248	**0.145**/**0.451**	0.102/0.414	0.094/0.414	0.092/0.447	0.204	0.634	0.206	0.991
	VAE	0.132/0.441	0.117/0.372	0.163/0.215	0.128/0.768	0.192/0.537	0.118/**0.854**	**0.494**/0.683	0.578	2.573	0.578	0.800
	ItemAE	0.029/**0.174**	0.010/0.027	0.000/0.084	0.004/0.166	0.002/0.157	0.000/0.147	**0.087**/0.162	0.524	3.000	0.537	0.931
	LightGCN	0.075/0.182	0.010/0.167	0.020/0.138	0.076/0.314	0.091/0.382	0.083/**0.403**	**0.266**/0.376	0.660	2.923	0.707	0.933
	NCF	0.275/0.541	0.208/0.462	0.164/0.491	0.197/0.827	0.262/**0.862**	0.232/0.817	**0.477**/0.835	0.553	1.735	0.571	0.969
Automotive	ItemCF	0.042/0.201	0.018/0.184	0.000/0.154	0.066/**0.324**	0.067/0.297	0.087/0.313	**0.204**/0.307	0.630	2.345	0.664	0.948
	WMF	0.019/0.286	0.027/0.439	0.000/0.337	0.082/0.294	0.044/0.438	0.018/0.398	**0.213**/**0.441**	0.483	2.598	0.483	1.005
	NGCF	0.010/0.124	0.000/0.145	0.001/0.119	0.000/0.096	0.051/**0.148**	**0.104**/0.140	0.087/0.123	0.588	0.836	0.707	0.831
	VAE	0.030/0.073	0.010/0.103	0.000/0.096	0.010/0.084	0.010/**0.172**	0.082/0.117	**0.091**/0.125	0.529	1.110	0.728	0.727
	ItemAE	0.020/0.320	0.010/0.176	0.002/0.208	0.020/0.321	0.008/0.311	0.091/0.310	**0.255**/**0.322**	0.792	2.802	0.792	1.003
	LightGCN	0.001/0.141	0.002/0.136	0.000/0.137	0.025/0.184	0.018/0.152	0.047/0.188	**0.162**/**0.191**	0.849	3.447	0.848	1.016
	NCF	0.082/0.503	0.070/0.515	0.002/0.544	0.112/0.808	0.104/0.762	0.142/0.774	**0.376**/**0.811**	0.464	2.648	0.464	1.004

only one fake user. We also list the results of multi-user injection in the right side of the slash for a comparison. We provide four types of ratio in Table 2 for better illustrating the results:

(1) **Ratio 1** indicates the percentage of the performance of SUI-Attack in single-user injection over the performance of the best baseline in multi-user injection. For example, the ratio 1 for ItemCF on T & HI is 0.194/0.295=0.658.
(2) **Ratio 2** represents the percentage of the single-user-injection performance of SUI-Attack over the single-user-injection performance of the best baseline. For example, the ratio 2 for ItemCF on T & HI is 0.194/0.042=4.620.
(3) **Ratio 3** shows the percentage of single-user-injection performance over multi-user-injection performance of SUI-Attack. For example, the ratio 3 for ItemCF on T & HI is 0.194/0.262=0.740.
(4) **Ratio 4** is the percentage of the performance of SUI-Attack in multi-user injection over the performance of the best baseline in multi-user injection. For example, the ratio 4 for ItemCF on T & HI is 0.262/0.295=0.888.

From Table 2, we have the following findings:

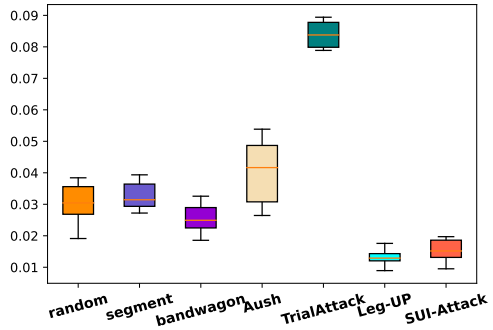
(1) Considering ratio 1, we can also see that SUI-Attack which injects only one fake user can generally achieve at least half of the attack performance of the best baseline in multi-user injection, and in some cases ratio 1 can even exceed 0.7. The observation is encouraging and we find that *even injecting only one fake user can severely mislead the RS to recommend the target item.* Hence, for some RS where defense mechanisms are deployed, SUI-Attack can effectively affect the RS without causing alarm.
(2) From ratio 2 shown in Table 2, we can observe that, in almost all cases, SUI-Attack outperforms baselines by a large margin for single-user injection, suggesting the superiority of SUI-Attack in single-user injection. The results also demonstrate that shilling attack methods without tailored designs for single-user injection cannot function well in this challenging setting.
(3) When more fake users are injected, we can find that the attack performance of SUI-Attack increases (see ratio 3 in Table 2) and SUI-Attack can achieve comparable or even better performance than contemporary shilling attacks (see ratio 4 in Table 2). Therefore, SUI-Attack, which is not specifically designed for the traditional shilling attack setting, can work well in multi-user injection, indicating its high flexibility.

4.3 Attack Invisibility

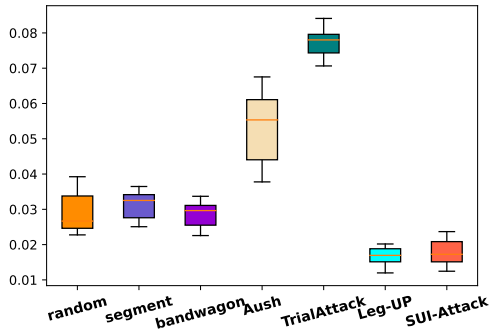
Compared to other shilling attack methods, SUI-Attack should be most difficult to detect as it only injects one fake user, the minimum injection for shilling attack, into the victim RS. Still, we investigate the invisibility of SUI-Attack following the study method used by existing shilling attack works [36, 37, 64] in this section.

4.3.1 Attack Detection. We use an unsupervised attack detector [71] to identify the fake user profiles generated by different attack models and report the precision and recall on Automotive in Figure 1. Since single-user injection is too difficult to detect, we report the detection results of multi-user injection for SUI-Attack. Lower precision and recall imply that the attack method is more imperceptible. The results show that, compared to other attack methods, it is more difficult to detect the fake users generated by SUI-Attack.

4.3.2 Fake User Distribution. To further study the invisibility of SUI-Attack, we visualize the users' representations using the t-SNE projection [52]. Note that we visualize the representation space in the multi-user injection as it is meaningless to visualize a single



(a) Precision



(b) Recall

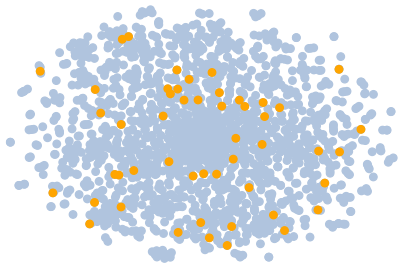
Figure 1: Attack detection of injected profiles on Automotive. Lower value suggests a better attack model.

fake user and many real users in single-user injection for checking whether they are different. Figure 2 provides the visualization of users' representations generated by WMF after it is attacked by SUI-Attack. We can observe that fake users are scattered among real users in the representation space and it is hard for detectors to distinguish fake and real users, suggesting that SUI-Attack can launch virtually invisible attacks.

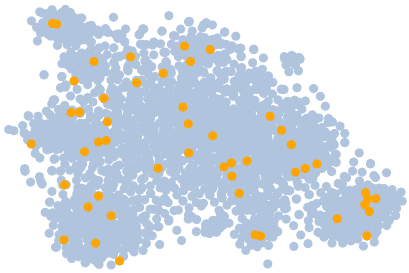
4.4 Ablation Study

Finally, we discuss the impact of different parts of SUI-Attack on the attack performance by conducting ablation experiments. Recall that our method mainly consists of two parts: feature generation and edge generation. The feature generation process also includes the influence function. Therefore, our ablation study involves three variants of SUI-Attack:

(1) **Replacing feature generation with random feature generation:** Randomly generated fake user features in SUI-Attack.
(2) **Replacing edge generation with random edge generation:** In edge generation, randomly connect the fake user node to other item nodes.
(3) **Removing the influence function:** It does not use the influence function to guide the generation of toxic fake user features.



(a) Last.fm



(b) Automotive

Figure 2: Real and fake user profiles in the latent space. Orange nodes represent injected fake users and other nodes are real users.

Figure 3: Ablation study.

Figure 3 shows the performance of the three variants of SUI-Attack compared to the performance of the complete SUI-Attack on Automotive. We can clearly see that the three variants show worse attack performance than SUI-Attack. Hence, we can conclude that each part in SUI-Attack contributes to its attack performance.

5 RELATED WORK

In this section, we introduce several directions that are closely related to this work.

5.1 Recommender Systems (RS)

The research on RS has a long history [1]. Traditional RS typically relies on collaborative filtering (CF) methods, especially matrix factorization (MF) [46], where user preferences and item properties are factorized from the user-item interaction matrix into two low-dimensional latent matrices. Due to its effectiveness on large-scale data [29], MF has been successfully deployed in practice. The cold-start problem (i.e., data sparsity), where historical data is not available for new users or items, is one of the most challenging issues in recommender systems [1]. To alleviate this problem, additional context features (e.g., social network [33, 34], user grouping data [12, 32], locations [38], sequential data [31], and review text [54]) are incorporated into MF.

Recently, the success of deep learning has inspired researchers to deploy deep learning in RS [59]. Various prevalent deep learning techniques have been applied in RS. For instance, RecSeats [40] adopts Convolutional Neural Network (CNN) in seat recommendation, Zhou et al. [72] uses Multilayer Perceptron (MLP) to enhance recommendation, Zhang et al. [70] deploy Recurrent Neural Network (RNN) to capture the local/global sessions within sequences for CTR prediction, Wang et al. [53] leverage Generative Adversarial Network (GAN) [17] in cross-domain recommendation, and Li et al. [30] harness Graph Neural Network (GNN) to model social recommendation. The use of deep learning methods has significantly improved the quality of recommendation [67].

Due to the importance of RS for guiding users towards making decisions, RS have attracted unscrupulous parties and there exist various types of attacks against RS in the literature, including unorganized malicious attacks (i.e., several attackers individually attack RS without an organizer) [42], sybil attacks (i.e., attacker illegally infers a user's preference) [6], shilling attack, etc.

5.2 Shilling Attack against RS

In the literature, shilling attack is also called as data poisoning attack [8, 28] or profile injection attack [5]. In experiments, previous works have successfully performed shilling attacks against real-world RS such as YouTube, Google Search, Amazon and Yelp [60, 61]. Sony, Amazon and eBay have also reported that they suffered from shilling attacks [27].

Pioneering shilling attack methods mainly rely on heuristics and data statistics. Lam and Riedl [27], Burke et al. [4, 5] and Mobasher et al. [39] propose several heuristic based shilling attack approaches to promote an item (e.g., Random, Average, Bandwagon and Segment Attacks) or demote an item (e.g., Love/Hate Attacks and Reverse Bandwagon Attacks) for both rating prediction and top-K recommendation. Wilson and Seminario [45, 57] propose power user attack and power item attack which leverage most influential users/items to hoax RS. Fang et al. [15] study shilling attack methods to spoof graph based RS. Li et al. [28] present shilling attack method against factorization based RS. Xing et al. [60] and Yang et al. [61] conduct experiments on attacking real-world RS (e.g., YouTube and Amazon), and the results show that attacking RS is possible in practice.

Recently, there is a surge of works on adversarial attack against text and image based learning systems [62, 68] and they show that, crafted adversarial examples, which may be imperceptible, can lead to unexpected mistakes of machine learning based systems. Based on the idea of adversarial attack, a great number of shilling attack approaches have sprung up. Optimization based methods [28, 50, 66] model shilling attacks as an optimization task and then design optimization strategies to solve it. GAN based methods [9, 36, 37, 58, 69] use GAN to construct fake user profiles. Reinforcement learning based methods [14, 48, 65] query the RS to get feedback on the attack. Then, they use Reinforcement Learning (RL) [23] to adjust the injection. Knowledge distillation based methods [63] and pre-training based methods [64] are designed to reduce the requirement of prior knowledge and improve the practicality of shilling attack.

Although many shilling attack methods exist, they all adopt the same attack paradigm, i.e., multi-user injection. None of them consider the extremely limited scenario, single-user injection, that studied in this work.

5.3 Adversarial Attacks in the Extremely Limited Scenarios

The idea of attacking a machine learning model by altering only one element of the input was first proposed in computer vision domain. Su et al [49] propose one-pixel attack and show that it can achieve high success rate when changing just one pixel to make the image misclassified by image classification algorithms. This work initiates the discussion of adversarial learning in extremely limited scenarios [2, 26]. Recently, Finkelshtein et al. [16] and Tao et al. [51] extend this idea to adversarial learning in graph representation learning. Finkelshtein et al. [16] shows that GNNs can be fooled by only slightly perturbing the features or the neighbor list of a single arbitrary node. The attack is effective even when the attacker cannot choose which node to perturb, and even when GNNs are trained with robust optimization techniques. Tao et al. [51] demonstrate that GNNs can be misled by a single injected node to misclassify the target node (i.e., single-node injection attack).

6 CONCLUSION

In this paper, we investigate a challenging scenario of shilling attack where only one fake user is injected into RS to launch the attack. We reformulate the shilling attack problem as a node generation task over the user-item bipartite graph of RS, which enables us to leverage more information in RS to construct the fake user profile. We propose SUI-Attack, the first shilling attack method that can be used in single-user injection. Experiments show that SUI-Attack can achieve promising attack results in single-user injection. Moreover, in the traditional multi-user injection setting, SUI-Attack is shown to be effective and can cause comparable attack results compared to existing shilling attack methods, showing its flexibility in shilling attacks. In the future, we will explore the underlying mechanism of the successful attack with a single injected node and try to design defense strategies against our SUI-Attack.

ACKNOWLEDGMENTS

This work was partially supported by National Key R&D Program of China (No. 2022ZD0118201), National Natural Science Foundation of China (No. 62002303, 42171456), and Natural Science Foundation of Fujian Province of China (No. 2020J05001).

REFERENCES

[1] Charu C. Aggarwal. 2016. *Recommender Systems - The Textbook.* Springer.

[2] Naveed Akhtar and Ajmal S. Mian. 2018. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* 6 (2018), 14410–14430.

[3] Al Borchers, Jonathan L. Herlocker, and John Riedl. 1998. Ganging up on Information Overload. *Computer* 31, 4 (1998), 106–108.

[4] Robin Burke, Bamshad Mobasher, and Runa Bhaumik. 2005. Limited Knowledge Shilling Attacks in Collaborative Filtering Systems. In *ITWP@IJCAI.*

[5] Robin D. Burke, Bamshad Mobasher, Runa Bhaumik, and Chad Williams. 2005. Segment-Based Injection Attacks against Collaborative Filtering Recommender Systems. In *ICDM.* 577–580.

[6] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. 2011. "You Might Also Like:" Privacy Risks of Collaborative Filtering. In *IEEE Symposium on Security and Privacy.* 231–246.

[7] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011). In *RecSys.* 387–388.

[8] Huiyuan Chen and Jing Li. 2019. Data Poisoning Attacks on Cross-domain Recommendation. In *CIKM.* 2177–2180.

[9] Konstantina Christakopoulou and Arindam Banerjee. 2019. Adversarial attacks on an oblivious recommender. In *RecSys.* 322–330.

[10] R Dennis Cook and Sanford Weisberg. 1980. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics* 22, 4 (1980), 495–508.

[11] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. 2022. A Survey on Adversarial Recommender Systems: From Attack/Defense Strategies to Generative Adversarial Networks. *ACM Comput. Surv.* 54, 2 (2022), 35:1–35:38.

[12] Danhao Ding, Hui Li, Zhipeng Huang, and Nikos Mamoulis. 2017. Efficient Fault-Tolerant Group Recommendation Using alpha-beta-core. In *CIKM.* 2047–2050.

[13] Pavlos S. Efraimidis and Paul G. Spirakis. 2006. Weighted random sampling with a reservoir. *Inf. Process. Lett.* 97, 5 (2006), 181–185.

[14] Wenqi Fan, Tyler Derr, Xiangyu Zhao, Yao Ma, Hui Liu, Jianping Wang, Jiliang Tang, and Qing Li. 2021. Attacking Black-box Recommendations via Copying Cross-domain User Profiles. In *ICDE.* 1583–1594.

[15] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. 2018. Poisoning Attacks to Graph-Based Recommender Systems. In *ACSAC.* 381–392.

[16] Ben Finkelshtein, Chaim Baskin, Evgenii Zheltonozhskii, and Uri Alon. 2022. Single-node attacks for fooling graph neural networks. *Neurocomputing* 513 (2022), 1–12.

[17] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS.* 2672–2680.

[18] Ihsan Gunes, Cihan Kaleli, Alper Bilge, and Huseyin Polat. 2014. Shilling attacks against recommender systems: a comprehensive survey. *Artif. Intell. Rev.* 42, 4 (2014), 767–799.

[19] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR.* 639–648.

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

[21] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM.* 263–272.

[22] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR (Poster).* https://openreview.net/pdf?id=rkE3y85ee

[23] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. 1996. Reinforcement Learning: A Survey. *J. Artif. Intell. Res.* 4 (1996), 237–285.

[24] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster).* https://arxiv.org/abs/1412.6980

[25] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *ICML*, Vol. 70. 1885–1894.

[26] David Kügler, Alexander Distergoft, Arjan Kuijper, and Anirban Mukhopadhyay. 2018. Exploring Adversarial Examples - Patterns of One-Pixel Attacks. In *MLCN/DLF/iMIMIC@MICCAI*, Vol. 11038. 70–78.

[27] Shyong K. Lam and John Riedl. 2004. Shilling recommender systems for fun and profit. In *WWW.* 393–402.

[28] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data Poisoning Attacks on Factorization-Based Collaborative Filtering. In *NIPS.* 1885–1893.

[29] Hui Li, Tsz Nam Chan, Man Lung Yiu, and Nikos Mamoulis. 2017. FEXIPRO: Fast and Exact Inner Product Retrieval in Recommender Systems. In *SIGMOD Conference.* 835–850.

[30] Hui Li, Lianyun Li, Guipeng Xv, Chen Lin, Ke Li, and Bingchuan Jiang. 2022. SPEX: A Generic Framework for Enhancing Neural Social Recommendation. *ACM Trans. Inf. Syst.* 40, 2 (2022), 37:1–37:33.

[31] Hui Li, Ye Liu, Nikos Mamoulis, and David S. Rosenblum. 2020. Translation-Based Sequential Recommendation for Complex Users on Sparse Data. *IEEE Trans. Knowl. Data Eng.* 32, 8 (2020), 1639–1651.

[32] Hui Li, Yu Liu, Yuqiu Qian, Nikos Mamoulis, Wenting Tu, and David W. Cheung. 2019. HHMF: hidden hierarchical matrix factorization for recommender systems. *Data Min. Knowl. Discov.* 33, 6 (2019), 1548–1582.

[33] Hui Li, Dingming Wu, and Nikos Mamoulis. 2014. A revisit to social network-based recommender systems. In *SIGIR.* 1239–1242.

[34] Hui Li, Dingming Wu, Wenbin Tang, and Nikos Mamoulis. 2015. Overlapping Community Regularization for Rating Prediction in Social Recommender Systems. In *RecSys.* 27–34.

[35] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW.* 689–698.

[36] Chen Lin, Si Chen, Hui Li, Yanghua Xiao, Lianyun Li, and Qian Yang. 2020. Attacking Recommender Systems with Augmented User Profiles. In *CIKM.* 855–864.

[37] Chen Lin, Si Chen, Meifang Zeng, Sheng Zhang, Min Gao, and Hui Li. 2022. Shilling Black-box Recommender Systems by Learning to Generate Fake User Profiles. *arXiv Preprint* (2022). https://arxiv.org/pdf/2206.11433.pdf

[38] Ziyu Lu, Hui Li, Nikos Mamoulis, and David W. Cheung. 2017. HBGG: a Hierarchical Bayesian Geographical Model for Group Recommendation. In *SDM.* 372–380.

[39] Bamshad Mobasher, Robin D. Burke, Runa Bhaumik, and Chad Williams. 2007. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans. Internet Techn.* 7, 4 (2007), 23.

[40] Théo Moins, Daniel Aloise, and Simon J. Blanchard. 2020. RecSeats: A Hybrid Convolutional Neural Network Choice Model for Seat Recommendations at Reserved Seating Venues. In *RecSys.* 309–317.

[41] Mohammad Amin Morid, Mehdi Shajari, and Ali Reza Hashemi. 2014. Defending recommender systems by influence analysis. *Inf. Retr.* 17, 2 (2014), 137–152.

[42] Ming Pang, Wei Gao, Min Tao, and Zhi-Hua Zhou. 2018. Unorganized Malicious Attacks Detection. In *NeurIPS.* 6976–6985.

[43] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW.* 285–295.

[44] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *WWW (Companion Volume).* 111–112.

[45] Carlos E. Seminario and David C. Wilson. 2014. Attacking item-based recommender systems with power items. In *RecSys.* 57–64.

[46] Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges. *ACM Comput. Surv.* 47, 1 (2014), 3:1–3:45.

[47] Mingdan Si and Qingshan Li. 2020. Shilling attacks against collaborative recommender systems: a review. *Artif. Intell. Rev.* 53, 1 (2020), 291–319.

[48] 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.* 157–168.

[49] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Trans. Evol. Comput.* 23, 5 (2019), 828–841.

[50] Jiaxi Tang, Hongyi Wen, and Ke Wang. 2020. Revisiting Adversarially Learned Injection Attacks Against Recommender Systems. In *RecSys.* 318–327.

[51] Shuchang Tao, Qi Cao, Huawei Shen, Junjie Huang, Yunfan Wu, and Xueqi Cheng. 2021. Single Node Injection Attack against Graph Neural Networks. In *CIKM.* 1794–1803.

[52] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* 9 (2008), 2579–2605.

[53] Cheng Wang, Mathias Niepert, and Hui Li. 2018. LRMM: Learning to Recommend with Missing Modalities. In *EMNLP.* 3360–3370.

[54] Cheng Wang, Mathias Niepert, and Hui Li. 2020. RecSys-DAN: Discriminative Adversarial Networks for Cross-Domain Recommender Systems. *IEEE Trans. Neural Networks Learn. Syst.* 31, 8 (2020), 2731–2740.

[55] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR.* 165–174.

[56] Chad Williams, Bamshad Mobasher, and Robin D. Burke. 2007. Defending recommender systems: detection of profile injection attacks. *Serv. Oriented Comput. Appl.* 1, 3 (2007), 157–170.

[57] David C. Wilson and Carlos E. Seminario. 2013. When power users attack: assessing impacts in collaborative recommender systems. In *RecSys.* 427–430.

[58] 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.

[59] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2023. A Survey on Accuracy-Oriented Neural Recommendation: From Collaborative Filtering to Information-Rich Recommendation. *IEEE Trans. Knowl. Data Eng.* 35, 5 (2023), 4425–4445.

[60] Xinyu Xing, Wei Meng, Dan Doozan, Alex C. Snoeren, Nick Feamster, and Wenke Lee. 2013. Take This Personally: Pollution Attacks on Personalized Services. In *USENIX Security Symposium.* 671–686.

[61] Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. 2017. Fake Co-visitation Injection Attacks to Recommender Systems. In *NDSS.*

[62] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial Examples: Attacks and Defenses for Deep Learning. *IEEE Trans. Neural Networks Learn. Syst.* 30, 9 (2019), 2805–2824.
[63] Zhenrui Yue, Zhankui He, Huimin Zeng, and Julian J. McAuley. 2021. Black-Box Attacks on Sequential Recommenders via Data-Free Model Extraction. In *RecSys*. 44–54.
[64] Meifang Zeng, Ke Li, Bingchuan Jiang, Liujuan Cao, and Hui Li. 2023. Practical Cross-System Shilling Attacks with Limited Access to Data. In *AAAI*. 4864–4874.
[65] Hengtong Zhang, Yaliang Li, Bolin Ding, and Jing Gao. 2020. Practical Data Poisoning Attack against Next-Item Recommendation. In *WWW*. 2458–2464.
[66] 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.
[67] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1 (2019), 5:1–5:38.
[68] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial Attacks on Deep-learning Models in Natural Language Processing: A Survey. *ACM Trans. Intell. Syst. Technol.* 11, 3 (2020).
[69] Xuxin Zhang, Jian Chen, Rui Zhang, Chen Wang, and Ling Liu. 2021. Attacking Recommender Systems With Plausible Profile. *IEEE Trans. Inf. Forensics Secur.* 16 (2021), 4788–4800.
[70] Xin Zhang, Zengmao Wang, and Bo Du. 2022. Deep Dynamic Interest Learning With Session Local and Global Consistency for Click-Through Rate Predictions. *IEEE Trans. Ind. Informatics* 18, 5 (2022), 3306–3315.
[71] Yongfeng Zhang, Yunzhi Tan, Min Zhang, Yiqun Liu, Tat-Seng Chua, and Shaoping Ma. 2015. Catch the Black Sheep: Unified Framework for Shilling Attack Detection Based on Fraudulent Action Propagation. In *IJCAI*. 2408–2414.
[72] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-enhanced MLP is All You Need for Sequential Recommendation. *arXiv Preprint* (2022). https://arxiv.org/abs/2202.13556