Shilling Black-Box Recommender Systems by Learning to Generate Fake User Profiles

Chen Lin, *Member, IEEE*, Si Chen, Meifang Zeng, Sheng Zhang, Min Gao, *Member, IEEE*, and Hui Li, *Member, IEEE*

***Abstract*— Due to the pivotal role of recommender systems (RS) in guiding customers toward the purchase, there is a natural motivation for unscrupulous parties to spoof RS for profits. In this article, we study shilling attacks where an adversarial party injects a number of fake user profiles for improper purposes. Conventional Shilling Attack approaches lack attack transferability (i.e., attacks are not effective on some victim RS models) and/or attack invisibility (i.e., injected profiles can be easily detected). To overcome these issues, we present learning to generate fake user profiles (Leg-UP), a novel attack model based on the generative adversarial network. Leg-UP learns user behavior patterns from real users in the sampled "templates" and constructs fake user profiles. To simulate real users, the generator in Leg-UP directly outputs discrete ratings. To enhance attack transferability, the parameters of the generator are optimized by maximizing the attack performance on a surrogate RS model. To improve attack invisibility, Leg-UP adopts a discriminator to guide the generator to generate undetectable fake user profiles. Experiments on benchmarks have shown that Leg-UP exceeds state-of-the-art shilling attack methods on a wide range of victim RS models. The source code of our work is available at: https://github.com/XMUDM/ShillingAttack.**

***Index Terms*— Black-box attack, generative adversarial network (GAN), recommender systems (RSs), shilling attack.**

I. Introduction

RECOMMENDER systems (RSs) have played a vital role since the beginning of e-commerce [1]. The prevalence of RS in industries (e.g., Amazon, Facebook, and Netflix [2]) has led to growing interest in manipulating RS [3]. On the one hand, unscrupulous parties can gain illegal profits by attacking RS to mislead users and affect their decisions. On the other hand, studying how to spoof RS gives insights into the defense against malicious attacks. We have seen various types of attacks against RS in the literature, including unorganized malicious attacks (i.e., several attackers individually attack RS without an organizer) [4], Sybil attacks (i.e., attacker illegally infers a user's preference) [5], and so on.

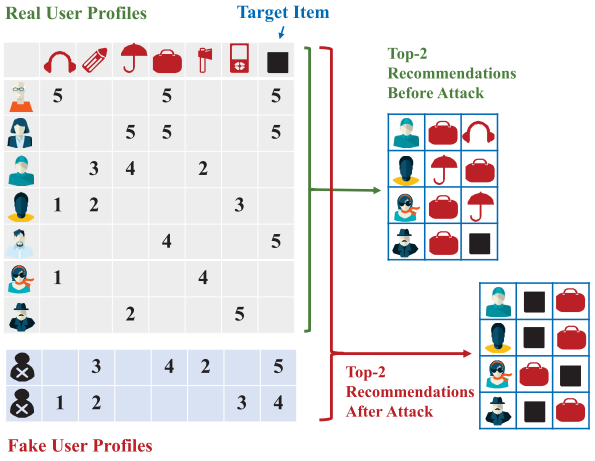

Fig. 1. Illustrative example of shilling attacks: promoting products by injecting fake user profiles. The value indicates the preference of a user on an item.

This article studies *shilling attack*, which is one of the most subsistent and profitable attacks against RS [6]. A shilling attack is also called a data poisoning attack [7], [8] or profile injection attack [9] in the literature. Researchers have successfully performed shilling attacks against real-world RS, such as YouTube, Google Search, Amazon, and Yelp, in experiments [10], [11]. Large companies like Sony, Amazon, and eBay have also reported that they suffered from shilling attacks in practice [12]. In shilling attacks, an adversarial party performs an attack by injecting a number of fake user profiles to hoax RS for improper purposes [12]. The goal is to either promote its own products (i.e., make the products recommended more often) or demote its competitors' products (i.e., make the competitors recommended less often). Fig. 1 illustrates the former case: after injecting carefully crafted fake user profiles, the target item appears in the top-two recommendation list provided by the victim recommendation model to some users. Shilling attacks fully utilize the core of RS: RS must allow users to interact with the system by various operations such as giving ratings and browsing pages. This way, RS can gain sufficient feedback from users to train their models and provide recommendations. The opening

Manuscript received 28 May 2021; revised 16 December 2021 and 22 March 2022; accepted 11 June 2022. The work of Chen Lin was supported in part by the National Natural Science Foundation of China under Grant 61972328, in part by the Alibaba Group through Alibaba Innovative Research Program, and in part by the Joint Innovation Research Program of Fujian Province of China under Grant 2020R0130. The work of Hui Li was supported in part by the National Natural Science Foundation of China under Grant 62002303, in part by the Natural Science Foundation of Fujian Province of China under Grant 2020J05001, and in part by the China Fundamental Research Funds for the Central Universities under Grant 20720210098. *(Corresponding author: Hui Li.)*

Chen Lin, Si Chen, Meifang Zeng, Sheng Zhang, and Hui Li are with the School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: chenlin@xmu.edu.cn; sichen@stu.xmu.edu.cn; zengmeifang@stu.xmu.edu.cn; zshingjason@163.com; hui@xmu.edu.cn).

Min Gao is with the School of Big Data and Software Engineering, Chongqing University, Chongqing 400044, China (e-mail: gaomin@cqu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3183210.

Digital Object Identifier 10.1109/TNNLS.2022.3183210

of RS to the data input from users makes injecting fake user profiles to launch shilling attacks possible. Moreover, shilling attacks treat the victim RS as a black box and only require the historical user-item interaction data (e.g., ratings) which is typically accessible from real users' public pages in the system. For instance, the pages of users containing their historical data in local business RS platforms Yelp[1] and Dianping[2] are both open to the public.

Early shilling attack methods create injection profiles based on simple heuristics [6], [12], [13]. For instance, average attack [12] assigns the highest rating to the target item to be promoted and average ratings to a set of randomly sampled items. Recently, with the success of adversarial attacks [14] in image classification [15] and text classification [16], a few works [17]–[20] consider directly adopting an adversarial attack model for shilling attacks. Though we have witnessed a great success of adversarial attacks against many learning systems, existing adversarial attacks cannot be directly adopted for shilling attacks.

1) *Attack Goals Are Different:* Adversarial attacks on text and image usually aim to trick the system to misclassify, while shilling attacks aim to misguide RS to rank the target item higher/lower.
2) *Attack Knowledge Is Different:* A prevalent strategy of adversarial attacks is to utilize the information of gradient descent in machine learning models to search undetectable perturbations [14]. However, in shilling attacks, though the data (e.g., rating matrix) of RS is generally available to all users (i.e., a user can see all other users' ratings) and thus exposed to attackers [6], [12], [13], the victim RS model is typically treated as a black box.
3) *Data Correlation Is Different:* Attacking many machine learning tasks can be achieved by manipulating one data sample (e.g., one-pixel attack [15]). This is impractical for RS since the recommendation for a user is typically made based on the information from multiple user-item pairs.

Besides, existing shilling attack methods suffer from the following limitations.

1) *Low Attack Transferability:* Simple heuristic-based attacking methods are shown to be effective only on certain traditional collaborative filtering (CF) approaches. For example, average [12], Bandwagon [21] and random [12] Attacks are more effective against CF using user-based k-nearest neighbors but do not work well against CF using item-based k-nearest neighbors [22]. Since they are tailored for one victim RS model, their attack transferability on other victim RS models, especially on recently prevalent deep learning-based RS models [23], is doubtful.
2) *Low Attack Invisibility:* Both simple heuristic-based and most recent methods which directly use adversarial attack models for shilling attacks lack the consideration of *personalization*. The generated fake user profiles do not have real-user behavior patterns in RS. Thus, the attack can be easily detected [18], [24]. Moreover, recent adversarial attack models for shilling attacks against RS [19], [20] generate fake user profiles with implicit feedback, i.e., profiles only have binary entries to indicate whether the user has interacted with an item. To launch shilling attacks using these "binary" fake user profiles, attackers have to use bots to automatically trigger interactions (e.g., frequently browse the page of an item) and mislead the system to regard the frequent operations as an indication of user preferences. Such frequent operations may arouse the suspicion of the detector.

In this article, we propose learning to generate fake user profiles (Leg-UP), which extends our previous work augmented shilling attack (AUSH) [24] for shilling attacks, and is built upon the generative adversarial network (GAN) [25]. The novelty of Leg-UP lies in the following designs to enhance its attack transferability and invisibility, which are different compared with AUSH and other recent shilling attack methods.

1) *Generator (Neural Network Module to Produce Discrete Ratings):* The generated fake user profiles of Leg-UP include discrete ratings. As giving Likert-scale ratings is the most common way that real RS provides to users to give feedback and get involved in the recommendation procedure, the fake user profiles are easy to inject into the system (e.g., type in ratings on a limited number of items manually), but difficult for the detector to discover. Leg-UP adopts a discretization layer (DL) in the generator which produces learnable thresholds to generate ratings with personalized user behavior patterns. To avoid the problem that discretization will encounter unpropagated gradients in training, we train Leg-UP through an approximation function. Since the error of the discretization will be eliminated during optimization, the generated fake user profiles are more accurate and powerful in shilling attacks. This way, both attack transferability (i.e., attack performance) and invisibility of Leg-UP are enhanced.
2) *Generator Optimization (Indirect and Direct Generation Losses):* The generator in a standard GAN is usually optimized through the classification task in the discriminator. Toward better attack performance, a *generation loss* can be associated solely with the generator. In AUSH, the generation loss is *indirect*, i.e., it is related to the reconstruction of some user data. Since the generator acts like an "attacker," the generation loss can also be *direct*, i.e., to measure how well the victim RS model is attacked. Inspired by Tang *et al.* [20], we design a generation loss to be estimated on a surrogate model for Leg-UP. We also propose a two-level optimization for the generation loss. Thus, less knowledge is demanded regarding the victim RS model, the black-box attack is allowed, and higher attack transferability can be achieved.
3) *Discriminator:* In GANs, the generator and the discriminator strike to enhance each other in a minimax competition. Specifically, in shilling attacks, by training the generator to "beat" the discriminator, the generated fake user profiles are less detectable and more powerful

[1]https://www.yelp.com
[2]http://www.dianping.com

in the attack. We provide a design of the discriminator for Leg-UP, which acts like a "defender" and identifies fake user profiles based on both explicit and implicit feedback so that both attack transferability and attack invisibility can be improved.

We have conducted comprehensive experiments to verify the effectiveness and undetectability of Leg-UP. We show that Leg-UP is effective against a wide range of victim RS models, including both classic and modern deep learning-based recommendation algorithms, on a variety of benchmark RS datasets. We also show that Leg-UP is virtually undetectable by the state-of-the-art attack detection method. We demonstrate, by visualizations, that the fake user profiles produced by Leg-UP follow a similar distribution as real users and maintain diversity (i.e., personalization). We hope that Leg-UP, as a new shilling attack method, can benefit the study of secure and robust RS.

The rest of this article is organized as follows: Section II surveys the related work. Section III introduces the background of shilling attacks against RS and the framework design of Leg-UP. Section IV and Section V describe in detail Leg-UP's model architecture and learning algorithm, respectively. In Section VI, we compare Leg-UP with other state-of-the-art shilling attack methods to verify its effectiveness and undetectability. Finally, Section VII concludes our work and points out future research directions.

II. Related Work

A. Recommender Systems

RS can help overcome the information overload problem. The research on RS has a long history. Early work on RS is either heuristic-based or factorization-based [26].

Recently, the great success of deep learning has significantly advanced the development of RS. Following are some representative deep neural networks adopted in RS and their examples: 1) convolutional neural network (CNN) can capture local and global representation from heterogeneous data [27]. One recent work RecSeats [28] adopts CNN to capture higher order interactions between features of available seats in seat recommendation. 2) Multilayer perceptron (MLP) can easily model the nonlinear interactions between users and items even when the system has much noise data. Recently, Zhou *et al.* [29] enhance MLP with filtering algorithms from signal processing that attenuates the noise in the frequency domain. Experimental results show that their method can significantly improve recommendation quality. 3) Recurrent neural network (RNN) can help RS model sequential behaviors better. For instance, Zhang *et al.* [30] recently adopt RNN in their proposed deep dynamic interest learning that captures the local/global sessions within sequences for CTR prediction and Xia *et al.* [31] leverage RNN for modeling reviews in RS. 4) Graph neural network (GNN) can model structural information in graph data that is prevalent in RS. Huang *et al.* [32] study how to use GNN together with negative sampling to provide recommendations on user-item graphs. Huang *et al.* [33] model both the high order user- and item-wise relation encoding in GNN for a social recommendation.

Due to the page limit, we only illustrate recent, representative RS. Readers can refer to related surveys [23], [26] for more related work on RS.

B. Adversarial Attacks

Investigating the security of machine learning-based systems is a continuing concern within the machine learning community. Research has shown that crafted adversarial examples [14], which may be imperceptible to the human eye, can lead to unexpected mistakes in machine learning-based systems. Adversarial attacks, which are launched by adversaries to leverage vulnerabilities of the system, have been studied in many text- and image-based learning systems [14], [34].

Adversarial examples in conventional machine learning models have been discussed for decades ago [14]. Dalvi *et al.* [35] find that manipulating input data may affect the prediction results of classification algorithms. Biggio *et al.* [36] design a gradient-based approach to generate adversarial examples against SVM. Barreno *et al.* [37], [38] formally investigate the security of conventional machine learning methods under adversarial attacks. Roli *et al.* [39] discuss several defense strategies against adversarial attacks to improve the security of machine learning algorithms. In addition to conventional machine learning, recent studies have reported that deep learning techniques are also vulnerable to adversarial attacks [14].

Even though adversarial attack methods are able to affect many learning applications, we cannot directly apply these methods to shilling attacks, as explained in Section I.

C. Generative Adversarial Network

GAN [25] performs adversarial learning between a generator and a discriminator, which can be implemented with any form of differentiable system that maps data from one space to the other. The generator tries to capture the real data distribution and generates real-like data, while the discriminator is responsible for discriminating between the generated and original data. GAN plays a minimax game and the optimization terminates at a saddle point that is a minimum with respect to the generator and a maximum with respect to the discriminator (i.e., Nash equilibrium).

As GAN overcomes the limitations of previous generative models [40], it has been prevalently deployed in applications that generate text [41], images [25], recommendations [42], or many other types of data [40]. To improve the original GAN, deep convolutional generative adversarial networks (DCGAN) [43] adopts the CNN architecture, and Wasserstein GAN [44] leverages Earth mover distance. There also exists a direction of GAN research that utilizes GAN to generate adversarial examples. For instance, Zhao *et al.* [45] propose to search the representation space of input data under the setting of GAN in order to generate more natural adversarial examples. Xiao *et al.* [46] design AdvGAN which can attack black-box models by training a distilled model.

D. Shilling Attacks Against RS

O'Mahony *et al.* [47], [48] first study the robustness of user-based CF by injecting some fake users. They also provide a theoretical analysis of the attack by viewing injected

ratings as noises. Lam and Riedl [12], Burke *et al.* [9], [21], and Mobasher *et al.* [22] further study the influence of some low-knowledge attack approaches to promote an item (e.g., random, average, bandwagon and segment attacks) and demote an item (e.g., love/hate attacks and reverse bandwagon attacks) on CF methods. Assuming more knowledge and cost are available, Wilson and Seminario [49] and Seminario and Wilson [50] design power user/item attacks which leverage the most influential users/items to shill RS, Fang *et al.* [51] study how to spoof graph-based RS models, and Li *et al.* [7] present near-optimal data poisoning attacks for factorization-based RS. Xing *et al.* [10] and Yang *et al.* [11] conduct experiments on attacking real-world RS (e.g., YouTube, Google Search, Amazon, and Yelp), and show that manipulating RS is possible in practice.

The success of adversarial attacks has inspired the study of shilling attacks. Christakopoulou and Banerjee [17], [18] employ DCGAN [43] to generate fake user profiles used in shilling attacks. However, directly adopting existing GANs will not provide satisfactory results in shilling attacks, as shown in our experiments. Tang *et al.* [20] leverage a surrogate victim model to estimate the attack performance for black-box attacks. Song *et al.* [19] design an effective shilling attack framework based on reinforcement learning. But their method requires that the feedback from RS is periodically available, which is impractical for most RS. Our previous work AUSH [24] is tailored for RS by considering attack cost and designing a specific generation loss. But AUSH applies simple rounding (SR) to generate discrete ratings, and its generation loss is indirect and only related to the reconstruction of some user data, which may affect the results of shilling attacks.

III. Background and Overview of Leg-UP

A. Terminology

First, we introduce the terminology of shilling attacks.

1) *Attack Goal:* The goal of an adversarial party in shilling attacks could be complex. In this article, we mainly consider targeted attack, i.e., one item must be influenced. This is the most common case for shilling attacks against RS because retail companies and producers have a strong intention to increase sales in the fierce business competition.
   a) **Push attacks** indicate that one or several target items must be promoted, i.e., the target items must be recommended by the victim RS model more than they were before the attack.
   b) **Nuke attacks** indicate that one or several target items must be demoted. Although we use push attacks to demonstrate the model design throughout this article, it is convenient to apply the techniques to nuke attacks by reversing the goal setting.
2) *Attack Budget:* Attacking RS is costly. When designing a practical shilling attack method against RS, we have to take into account the attack budgets from two perspectives:
   a) **Attack size** is the number of fake user profiles. The larger the attack size is, the more effective and expensive the attack could be.
   b) **Profile size** is the number of nonzero ratings in one fake user profile. Some recent works [20] do not consider profile size. However, we believe that constraining the profile size is necessary. In some e-commerce platforms, injecting fake ratings is impossible without real purchases. Thus, it will be too expensive to generate fake users with unlimited nonzero ratings.
3) *Attack Knowledge:* How much knowledge is accessible to the attacker is a critical factor in designing shilling attack methods. In general, the most desirable knowledge is related to the user feedback and the victim RS model.
   a) **User feedback** is the dataset used to train the victim RS model. The attacker can have full or partial knowledge about the user feedback. In this article, we assume that the attacker has full knowledge of the user feedback, i.e., the attacker knows who rates what and the exact rating values. This is a reasonable setting and is commonly adopted in literature [6], [12], [13], [20] because user-item ratings in RS are generally available to all users (e.g., a user can see all other users' ratings) and thus exposed to attackers.
   b) **Victim RS model** is used to deliver recommendations and it is the target of shilling attacks. Some existing works assume that a white-box attack is possible, i.e., the attacker has different degrees of knowledge about the victim model, e.g., the model type [51] and model parameters [52]. In this article, we assume that the victim model is unseen to the attacker (i.e., a black box). This is because, in many real productive RS, the model is so complex that acquiring extraordinary knowledge about the model parameters is impossible. Furthermore, real RS usually employ ensemble models and update their models frequently [1]. Therefore, the attacker cannot have accurate knowledge about the model type.
4) *Injected Fake User Profiles:* Although Leg-UP does not explicitly classify items in a user profile, we follow the terminology used in the literature [6] and divide the items in a fake user profile into four parts so that our descriptions are consistent with previous works.
   a) **Target item** indicates the item that the attacker wants to fulfill his/her malicious purpose.
   b) **Filler items** are the items that have nonzero ratings in the injected fake user profiles. The filler items in each fake user profile are usually different.
   c) **Unrated items** are the items that have not been assigned with any ratings in the injected fake user profiles.
   d) **Selected items** are several human-selected items for a special treatment. Not all attack models consider selected items. One possible reason to use selected items is to influence *in-segment users*, i.e., users that have shown preferences on selected items. Details can be found in Section VI-E.

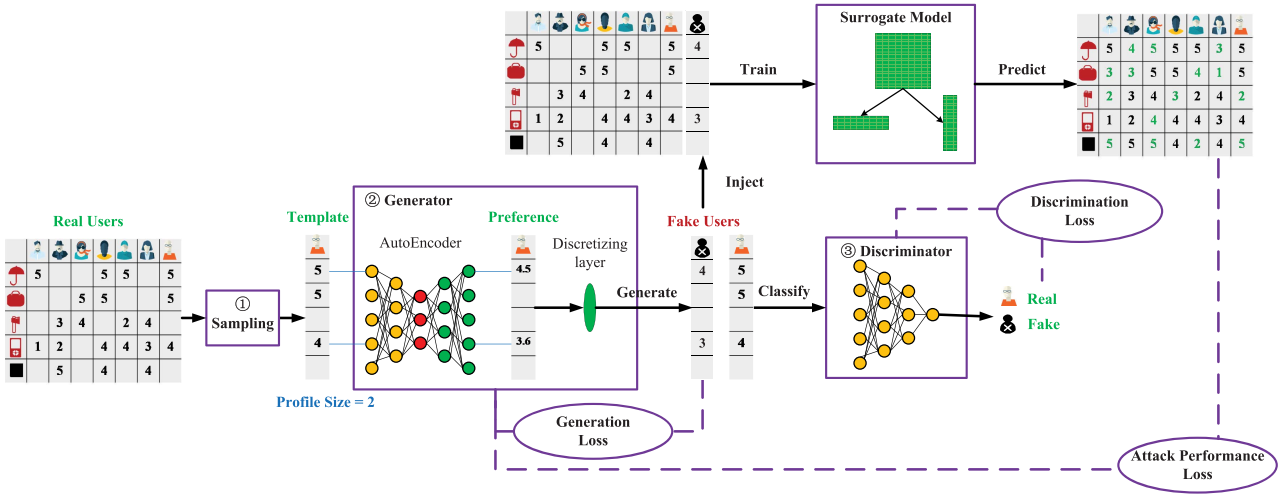

Fig. 2. Overview of Leg-UP. Leg-UP consists of three parts: 1) sampling step that samples real user profiles as "templates;" 2) generator that generates fake users; and 3) discriminator that distinguishes real user profiles and fake user profiles to boost the generator's ability to generate undetectable fake user profiles.

TABLE I
NOTATIONS FOR LEG-UP

Notation	Definition
$\mathcal{U}, \mathcal{V}, \mathcal{I}, \mathcal{S}$	Sets of real users, fake users, items, and selected items.
$\mathbf{X} \in \mathcal{R}^{\lvert\mathcal{U}\rvert \times \lvert\mathcal{I}\rvert}$	Input rating matrix of real users and items.
$G_\Theta(\mathbf{X}^{(in)})$ or $\hat{\mathbf{X}} \in \mathcal{R}^{\lvert\mathcal{V}\rvert \times \lvert\mathcal{I}\rvert}$	Injected fake user rating matrix.
$\mathbf{X}^{(in)} \in \mathcal{R}^{A \times \lvert\mathcal{I}\rvert}$	Sampled user rating matrix used for "templates".
$\tilde{\mathbf{X}} \in \mathcal{R}^{* \times *}$	RS's predictions on some users and items.
$\mathbf{X}_{u,:}$	A row of $\mathbf{X}$ represents a user u.
$\mathbf{X}_{:,i}$	A column of $\mathbf{X}$ represents an item i.
$\mathcal{I}_u = \{i \in \mathcal{I} : \mathbf{X}_{u,i} \neq 0\}$	The item set that user u has rated.
$\mathcal{U}_i = \{u \in \mathcal{U} : \mathbf{X}_{u,i} \neq 0\}$	The set of users who have rated item i.
$\hat{\mathbf{X}} = G_\Theta(\mathbf{X}^{(in)})$	The generator producing fake user profiles based on "templates", parameterized by Θ.
$D_\Phi(\hat{\mathbf{X}})$	The discriminator parameterized by Φ.
$O_\Gamma(\mathbf{X}, \hat{\mathbf{X}})$	The surrogate victim RS model parameterized by Γ.

B. Notations

Throughout this article, we use lowercase letters for indices, capital letters for scalars, boldface lower-case letters for vectors, boldface capital letters for matrices, and calligraphic letters for sets. The notations used for Leg-UP are illustrated in Table I. We use $\mathbf{X} \in \mathcal{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ to denote the real-rating matrix in RS, where $\mathcal{U}$ and $\mathcal{I}$ represent the user universe and the item universe, respectively. Each row of $\mathbf{X}$, i.e., $\mathbf{X}_{u,:}$ represents the ratings given by user u. Each column of $\mathbf{X}$, i.e., $\mathbf{X}_{:,i}$ represents the ratings assigned to item i. $\mathcal{I}_u = \{i \in \mathcal{I} : \mathbf{X}_{u,i} \neq 0\}$ indicates the set of items that have been rated by user u. Similarly, $\mathcal{U}_i = \{u \in \mathcal{U} : \mathbf{X}_{u,i} \neq 0\}$ denotes the set of users that have rated item i. The attack budget is given by A and P, i.e., the attack size and profile size. Leg-UP takes $\mathbf{X}$ as the input and generates the fake user profiles $\hat{\mathbf{X}}$, where $\hat{\mathbf{X}} \in \mathcal{R}^{|\mathcal{V}|\times|\mathcal{I}|}$. Since the attack size is A, $|\mathcal{V}| = A$. Given the profile size P, there are exactly P nonzero entries in $\hat{\mathbf{X}}_{v,:}$, $\forall v \in \mathcal{V}$.

C. Framework Overview

Inspired by the success of GAN-based adversarial learning in text [41] and image [25] generation, the design of Leg-UP follows a GAN framework. Fig. 2 gives an overview of our pipeline, which consists of the following parts.

1) *Sampling:* Making up fake user profiles from scratch is risky as the generated profiles may not have real-user patterns. Following our previous work AUSH [24], we use sampled real user profiles as "templates" and learn real user patterns from them. Using "templates" instead of all the real users for learning real user patterns is computationally efficient and makes it easy for Leg-UP to focus on capturing patterns of real users with rich information which leads to high-quality fake profiles. The knowledge of templates is accessible in practice and does not exceed the requirements of recent sophisticated attack methods [7], [51], as we have shown in Section III-A. The sampling component in Leg-UP gives the generator the opportunity to learn from real user behaviors and retain the preference diversity of the community (i.e., personalization). Especially, the sampling component contains two steps. In the first step, a batch of real users is chosen as "templates." The second step is optional. When a fixed profile size is required, in the second step, filler items are sampled from the rated items of the corresponding "template."
2) *Generator:* The generator takes as input the sampled user-item rating submatrix (i.e., "templates") and captures the latent association between items and users. It outputs the injected fake user profiles in the form of discrete ratings. To boost the attack performance, the parameters of the generator are learned by minimizing a *generation loss*, which measures how much the victim RS model will be affected after the attack. As we are performing black-box attacks, the details of the victim recommendation model are unknown. Thus, the generation loss is measured via a surrogate victim model in Leg-UP.
3) *Discriminator:* The discriminator of Leg-UP is fed with the output of the generator. It distinguishes real user profiles and fake user profiles. Optimizing the discriminator,

through a *discrimination loss* which measures the accuracy of the classification of real/fake user profiles, boosts the generator's ability to generate undetectable fake user profiles.

The design of Leg-UP is general and there are various possible implementations for the generator and the discriminator. We provide the details of one implementation in Section IV.

IV. Model

In this section, we will elaborate on one implementation for Leg-UP to accomplish shilling attacks.

A. Sampling

As illustrated in Section III-C, Leg-UP contains a two-step sampling component.

1) *User Sampling:* The first step is to sample users $u \sim \mathcal{U}$ from the existing users to construct "templates." The result is a submatrix of $\mathbf{X}$, i.e., $\mathbf{X}^{(\text{in})} \in \mathcal{R}^{|\mathcal{U}'|\times|\mathcal{I}|}$. Given the attack size A, we have $|\mathcal{U}'| = A$. Each "template" is sampled randomly from real users.
2) *Filler Item Sampling:* Given the profile size P, Leg-UP uses the second step to select items in each "template." We follow the setting of AUSH [24] and randomly keep P items if the sampled user (i.e., "template") has interacted with more than P items. If the sampled user has interacted with less than P items, we only use these items in this "template."

B. Generator

The generator aims to capture the real user preferences from "templates" in order to construct the fake user profiles in the form of discrete ratings for shilling attacks. There are various possible model structures for the generator. Without loss of generality, we divide the generator of Leg-UP into two major components. One is used to generate user preferences (i.e., numerical values in the latent space) from "templates," and the other is to transform the user preferences into discrete ratings.

1) Generating User Preference: To infer user preferences in $\mathbf{X}^{(\text{in})}$, a function, *preference learner*, that yields numerical output is required. Suppose the preference learner is denoted by $\text{PL}_\Omega(\mathbf{X}^{(\text{in})})$, with parameters $\Omega = \{\omega\}$. The simplest preference learner is to directly output its parameters

$$\text{PL}_\Omega(\mathbf{X}^{(\text{in})})_{v,i} = \omega_i. \tag{1}$$

We can also adopt a more complex neural network module to extract user preferences from "templates." One attractive feature of using such a module is that it can capture nonlinear user-item associations in the "templates." Since most victim RS models deliver recommendations based on user-item correlations, adopting a complex preference learner can be helpful in shilling the victim RS model. To be specific, we empirically find that the following autoencoder (AE)-based design works well for Leg-UP :

$$\begin{aligned}\text{PL}_\Omega(\mathbf{X}^{(\text{in})})_v &= f_{\text{decode}}\left(f_{\text{encode}}\left(\mathbf{X}^{(\text{in})}_{v,:}\right)\right) * 2.5 + 2.5\\ f_{\text{encode}}(\mathbf{X}) &= \text{ReLU}(\mathbf{W}_e\mathbf{X} + \mathbf{b}_e)\\ f_{\text{decode}}(\mathbf{X}) &= \text{Tanh}(\mathbf{W}_d\mathbf{X} + \mathbf{b}_d)\end{aligned} \tag{2}$$

where $\mathbf{W}$ and $\mathbf{b}$ are learnable weight matrices and bias vectors, respectively. In (2), $f_{\text{encode}}(\mathbf{X}^{(\text{in})}_{v,:})$ is a MLP which transforms the ratings $\mathbf{X}^{(\text{in})}_{v,:}$ into a low-dimensional latent representation, i.e., user preferences. $f_{\text{decode}}(\cdot)$ is another MLP that decodes the latent representation back to a rating vector. Note that, in the implementation, each template has at most P nonzero entries, i.e., $\|\mathbf{X}^{(\text{in})}_{v,:}\|_0 \leq P$ and only the nonzero entries are involved in computing. To ensure that fake user profiles will have exactly P nonzero ratings, we operate on the corresponding entries of the output, hence, we have $\|\text{PL}_\Omega(\mathbf{X}^{(\text{in})})_v\|_0 \leq P$. Moreover, we normalize the output of the preference learner via first multiplying it by 2.5 and then adding 2.5 so that $0 < \text{PL}_\Omega(\mathbf{X}^{(\text{in})})_v < 1$.

2) Discretizing User Preference: Giving discrete ratings (e.g., five-star scale ratings) to items is the common way that RS provides to users to interact with the system and get involved in the recommendation procedure. Thus, the numerical output of AE in Leg-UP, which indicates the preference strength of each fake user on each filler item, is designed to be discretized as five-star ratings. Our previous work AUSH [24] adopts a simple rounding method to project numerical preference to discrete ratings. The rounding method in AUSH divides the range into five segments

$$\hat{\mathbf{X}}_{v,i} = \begin{cases} 1, & 0 < \text{PL}_\Omega(\mathbf{X}^{(\text{in})})_{v,i} \leq 0.2\\ 2, & 0.2 < \text{PL}_\Omega(\mathbf{X}^{(\text{in})})_{v,i} \leq 0.4\\ 3, & 0.4 < \text{PL}_\Omega(\mathbf{X}^{(\text{in})})_{v,i} \leq 0.6\\ 4, & 0.6 < \text{PL}_\Omega(\mathbf{X}^{(\text{in})})_{v,i} \leq 0.8\\ 5, & 0.8 < \text{PL}_\Omega(\mathbf{X}^{(\text{in})})_{v,i} < 1. \end{cases} \tag{3}$$

However, the rounding method in AUSH is intuitively suboptimal.

1) It cannot discover subtle differences among users as the rounding method is identical to all users and/or items. For example, suppose the estimated preference is 0.85, a normal user may give a five-star rating, while a picky user might give a four-star rating. Rounding will output five-star for all users, which is not feasible.
2) It will incur unpredictable errors that influence attack performance. Since rounding is performed when the optimization of AUSH is completed, errors introduced by discretizing the preference are not accumulated in the learning objective. Suppose that the estimated preference from AUSH is 4.8, which has minimized the generation loss of AUSH. Then, discretizing it to 5 may not conform to the direction of optimization for the generation loss, which will affect the attack performance.

To address the above-mentioned issues in AUSH, we design a final DL in the generator of Leg-UP to discretize preference so that the discretization process becomes part of the optimization and discretization errors can be reduced

$$G_\Theta(\mathbf{X}^{(\text{in})})_{v,i} = \sum_{k=1}^{5} H\left(\text{PL}_\Omega(\mathbf{X}^{(\text{in})})_{v,i}, \sum_{k'=1}^{k}\tau_{v,k'}\right) \tag{4}$$

where τ is a learnable parameter, $\forall v \in \mathcal{V}, 0 \leq \tau_{v,k} \leq 1$. $\Theta = \{\tau, \mathbf{W}, \mathbf{b}\}$ is the set of parameters of AE and the final DL. $H(x, \tau), 0 < x < 1$ is the Heaviside step function defined

as follows:

$$H(x,\tau)=\begin{cases}1, & x>\tau\\ 0, & x\le\tau.\end{cases} \tag{5}$$

Essentially, the final DL learns five value segments for each user v

$$\hat{\mathbf{X}}_{v,i}=\begin{cases}1, & 0<\mathrm{PL}_{\Omega}(\mathbf{X}^{(\mathrm{in})})_{v,i}\le\tau_{v,1}\\ 2, & \tau_{v,1}<\mathrm{PL}_{\Omega}(\mathbf{X}^{(\mathrm{in})})_{v,i}\le\sum_{k=1}^{2}\tau_{v,k}\\ 3, & \tau_{v,1}+\tau_{v,2}<\mathrm{PL}_{\Omega}(\mathbf{X}^{(\mathrm{in})})_{v,i}\le\sum_{k=1}^{3}\tau_{v,k}\\ 4, & \sum_{k=1}^{3}\tau_{v,k}<\mathrm{PL}_{\Omega}(\mathbf{X}^{(\mathrm{in})})_{v,i}\le\sum_{k=1}^{4}\tau_{v,k}\\ 5, & \sum_{k=1}^{4}\tau_{v,k}<\mathrm{PL}_{\Omega}(\mathbf{X}^{(\mathrm{in})})_{v,i}<\sum_{k=1}^{5}\tau_{v,k}.\end{cases} \tag{6}$$

We can attach a generation loss with the generator to enhance the quality of generated fake user profiles. It is worth noting that, even without the generation loss, the generator can still get rewarded/penalized through the discrimination loss introduced in Section IV-C. Generation loss can be *indirect*. For example, AUSH employs a *reconstruction loss* which measures how well the generated ratings recover the real ratings

$$\mathcal{L}_{\mathrm{gen}}=\sum_{v\in\mathcal{U}}\sum_{\mathbf{X}^{(\mathrm{in})}_{v,j}\neq 0}\left(\hat{\mathbf{X}}_{v,j}-\mathbf{X}^{(\mathrm{in})}_{v,j}\right)^2. \tag{7}$$

Equation (7) does not require any prior knowledge of the victim RS model and it is irrelevant to the attack. The design goal of the reconstruction loss is to help the fake user profiles retain the real user behavior patterns so that they can be difficult to detect.

We can also *directly* maximize the attack performance via the generation loss. We use push attacks as an example, i.e., how much the target item t will be promoted

$$\mathcal{L}_{\mathrm{gen}}=-\sum_{u\in\mathcal{U}}\log\frac{\exp\tilde{\mathbf{X}}_{u,t}}{\sum_{j\in\mathcal{I}}\exp\tilde{\mathbf{X}}_{u,j}} \tag{8}$$

where $\tilde{\mathbf{X}}_{u,i}$ is the rating predicted by the victim model for user u on item i. If the target item t receive a higher predicted rating by the victim RS model than other items after the attack, then the direct loss in (8) will be minimized. Note that nuke attacks can be achieved similarly via maximizing (8).

However, we cannot obtain the predicted ratings $\tilde{\mathbf{X}}$ by the victim RS model in the setting of the black-box shilling attacks. Instead, we use the output of a surrogate RS model to represent $\tilde{\mathbf{X}}$, with the assumption that the knowledge for attacking a well-trained surrogate RS model can be transferred to attacking other RS models (i.e., real-victim RS models) [20]. Therefore, the predicted ratings $\tilde{\mathbf{X}}$ can be obtained via

$$\begin{aligned}\tilde{\mathbf{X}}&=S_{\Gamma}\left(\mathbf{X}^{*}\right)\\ \mathbf{X}^{*}&=\mathrm{concate}\left(\mathbf{X},G_{\Theta}(\mathbf{X}^{(\mathrm{in})})\right)\end{aligned} \tag{9}$$

where S_{Γ} indicates a surrogate RS model with parameters Γ, and $\mathbf{X}^{*}$ indicates the concatenation of two matrices $\mathbf{X}$ and $G_{\Theta}(\mathbf{X}^{(\mathrm{in})})$. Note that the surrogate model is trained on both real user feedback and fake user profiles to achieve a minimal recommendation loss $\mathcal{L}_{\mathrm{RS}}$

$$\Gamma=\arg\min_{\Gamma}\ \mathcal{L}_{\mathrm{RS}}\left(\mathbf{X}^{*},S_{\Gamma}(\mathbf{X}^{*})\right). \tag{10}$$

We use the original design of the recommendation loss $\mathcal{L}_{\mathrm{RS}}$ in the surrogate RS. But we calculate $\mathcal{L}_{\mathrm{RS}}$ on $\mathbf{X}^{*}$ instead of $\mathbf{X}$.

In summary, we have the following objective for the generator:

$$\begin{aligned}\min_{\Theta}\ & -\sum_{u\in\mathcal{U}}\log\frac{\exp S_{\Gamma}\left(\mathbf{X}^{*}\right)_{u,t}}{\sum_{j\in\mathcal{I}}\exp S_{\Gamma}\left(\mathbf{X}^{*}\right)_{u,j}}\\ \text{s.t.},\ & \Gamma=\arg\min\ \mathcal{L}_{\mathrm{RS}}\left(\mathbf{X}^{*},S_{\Gamma}(\mathbf{X}^{*})\right).\end{aligned} \tag{11}$$

The choice of the surrogate model is important to ensure attack transferability. Following Tang *et al.* [20], we use weighted regularized matrix factorization (WRMF) as the default surrogate model due to its efficiency and effectiveness on real data [1], [2]. But we also explore other surrogate models in our experiments in Section VI-G. In WRMF, the recommendation loss $\mathcal{L}_{\mathrm{RS}}$ is defined as the weighted aggregation of the differences between predictions and observed ratings in $\mathbf{X}^{*}$

$$\begin{aligned}\mathcal{L}_{\mathrm{RS}}\left(\mathbf{X}^{*},S_{\Gamma}(\mathbf{X}^{*})\right)=&\sum_{u\in\mathcal{U},i\in\mathcal{I}}w_{u,i}\left(\mathbf{X}^{*}_{u,i}-\mathbf{P}_{u}^{T}\mathbf{Q}_{i}\right)^2\\ &+\lambda(\|\mathbf{P}\|^2+\|\mathbf{Q}\|^2)\end{aligned} \tag{12}$$

where $w_{u,v}$ is the instance weight for the observed rating $\mathbf{X}_{u,i}\neq 0$ and the missing rating $\mathbf{X}_{u,i}=0$. $\Gamma=\{\mathbf{P},\mathbf{Q}\}$ are model parameters of the surrogate model, and λ is the hyperparameter to control model complexity.

C. Discriminator

The discriminator D attempts to correctly distinguish fake user profiles and real user profiles. It takes the fake user profiles generated by the generator or the real user profiles and outputs the probabilities that the inputs are real. We use an MLP as our discriminator. For simplicity, we illustrate the discriminator in (13) with the fake user profiles $\hat{\mathbf{X}}$

$$D_{\Phi}(\hat{\mathbf{X}})=\sigma\left(\mathbf{W}_{d}(\hat{\mathbf{X}})+\mathbf{b}_{d}\right) \tag{13}$$

where $\Phi=\{\mathbf{W}_{d},\mathbf{b}_{d}\}$ are learnable weight matrices and bias vector, and σ indicates the sigmoid function. Note that we use both explicit feedback (nonzero ratings) and implicit feedback (denoted as zeros) in the user profiles for (13).

The goal of the discriminator is essentially a binary classification task. We use the cross-entropy loss as the discrimination loss

$$\mathcal{L}_{\mathrm{dis}}=\log D_{\Phi}(\mathbf{X})+\log\left(1-D_{\Phi}(\hat{\mathbf{X}})\right). \tag{14}$$

Inspired by the idea of GAN [25], we aim to unify the different goals of the generator and the discriminator by letting them play a *minimax game* via optimizing the objective in (15).

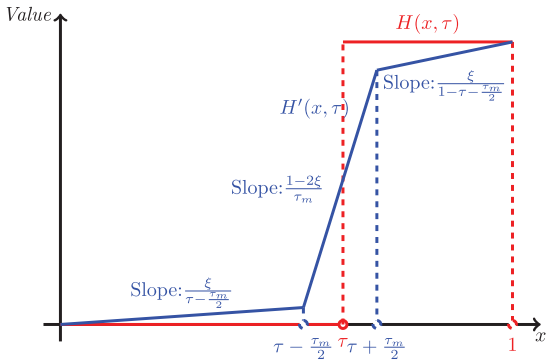

Fig. 3. We use $H'(x,\tau)$ (line in blue) to approximate the Heaviside step function $H(x,\tau)$ (line in red). The approximation contains three straight-line segments with different slope.

Using the direct generation loss as an example, the overall objective of Leg-UP is presented as follows:

$$\begin{aligned}\min_{\Theta}\max_{\Phi}\ \mathcal{L} &= \mathbb{E}_{u\sim\mathcal{U}}[\log D_\Phi(\mathbf{X}_u)] \\ &\quad - \sum_{u\in\mathcal{U}} \log \frac{\exp\left(S_\Gamma(\mathbf{X}^*)_{u,t}\right)}{\sum_{j\in\mathcal{I}}\exp\left(S_\Gamma(\mathbf{X}^*)_{u,j}\right)} \\ &\quad + \mathbb{E}_{\hat{\mathbf{X}}_v\sim G_\Theta(\mathbf{X}^{(\text{in})})}\left[\log\left(1-D_\Phi(\hat{\mathbf{X}}_v)\right)\right] \\ \text{s.t., } \Gamma &= \arg\min\ \mathcal{L}_{\text{RS}}\left(\mathbf{X}^*, S_\Gamma(\mathbf{X}^*)\right)\end{aligned} \tag{15}$$

where Θ and Φ are model parameters of G and D, respectively. u is a real user profile sampled from the user universe, $\hat{\mathbf{X}}_v$ is a fake user profile generated from the generator distribution $G_\Theta(\mathbf{X}^{(\text{in})})$.

V. Learning

Training Leg-UP is not trivial. We will discuss and provide solutions to three critical problems in training Leg-UP in this section.

A. Approximate Discretization

First, the DL in the generator employs a discontinuous step function. The gradient for the step function is not defined at the boundary and is zero everywhere else. This is problematic for gradient propagation.

To address this issue, we adopt an approximation function H' to mimic the behavior of the Heaviside step function H introduced in (5)

$$H'(x,\tau) = \begin{cases} \dfrac{x\xi}{\tau-\frac{\tau_m}{2}}, & x < \tau-\dfrac{\tau_m}{2} \\ \dfrac{\xi\left(x-\tau-\frac{\tau_m}{2}\right)}{1=\tau-\frac{\tau_m}{2}}+1-\tau, & x > \tau+\dfrac{\tau_m}{2} \\ \dfrac{(x-\tau)(1-2\xi))}{\tau_m}+0.5, & \text{others} \end{cases} \tag{16}$$

where $\tau_m = \min\{\tau, 1-\tau\}$. As shown in Fig. 3, $H'(x,\tau)$ uses three straight lines to approximate $H(x,\tau)$, when $x\in[0,1]$. As explained by Tsoi *et al.* [53], a larger value of ξ providers a smoother derivative but further approximation from the Heaviside function. In practice, a value of $\xi = 0.1$ works well.

Algorithm 1 Training Procedure for Leg-UP

Input: rating matrix $\mathbf{X}$
Output: parameter set Θ for the generator G and parameter set Φ for the discriminator D
for *number of training epochs* **do**
  (1) Discriminator optimization
  for k_1 *steps* **do**
    uniformly sample a minibatch of users $\mathcal{U}'$;
    foreach $u'\in\mathcal{U}'$ **do**
      sample F items to construct $\mathbf{x}_{u'}^{(in)}$;
    generate a minibatch of fake user profiles $\{\mathbf{x}_{u'}^{(out)} = G(\mathbf{x}_{u'}^{(in)})|u'\in\mathcal{U}'\}$;
    optimize Φ to min $\mathcal{L}_{\text{dis}}$ with Θ fixed;
  (2) Generator optimization
  for k_2 *steps* **do**
    uniformly sample a minibatch of user rating vectors $\{\mathbf{x}_u\}$;
    foreach $u'\in\mathcal{U}'$ **do**
      sample F items to construct $\mathbf{x}_{u'}^{(in)}$;
    generate a minibatch of fake user profiles $\{\mathbf{x}_{u'}^{(out)} = G(\mathbf{x}_{u'}^{(in)})|u'\in\mathcal{U}'\}$;
    (3) Surrogate optimization
    for T *steps* **do**
      sample a minibatch of user rating vectors from $(\mathbf{X},\hat{\mathbf{X}})$;
      optimize Γ to min $\mathcal{L}_{\text{RS}}$;
    optimize Θ to min $\mathcal{L}_{\text{gen}}$ with Φ and Γ fixed;

B. Two-Level Optimization for $\mathcal{L}_{\text{gen}}$

Second, the adversarial loss in (11) is computed based on a well-trained surrogate RS model, which requires a two-level optimization procedure. The exact gradient, with respect to the generator parameters, for the adversarial loss can be written as

$$\partial_\Theta\mathcal{L}_{\text{gen}} = \frac{\partial\mathcal{L}_{\text{gen}}}{\partial\hat{\mathbf{X}}}\frac{\partial\hat{\mathbf{X}}}{\partial\Theta} + \frac{\partial\mathcal{L}_{\text{gen}}}{\partial\Gamma}\frac{\partial\Gamma}{\partial\hat{\mathbf{X}}} \tag{17}$$

where $\hat{\mathbf{X}} = G_\Theta(\mathbf{X})$.

However, the computation of the exact gradient requires keeping the parameters of the surrogate model for all the training steps, which will consume a lot of time and space resources [20]. Following the idea of Tang *et al.* [20], we approximate the gradient by unrolling only the last step when accumulating gradients to improve the efficiency of Leg-UP.

C. Learning Algorithm

Finally, we present Algorithm 1 to optimize the overall objective of Leg-UP in (15). Specifically, the generator acts like an "attacker" and attempts to generate "perfect" fake user profiles that are difficult to detect by the discriminator and can achieve the malicious attack goal on the surrogate model. On the other hand, the discriminator module performs like a "defender." It tries to accurately distinguish fake user profiles and real user profiles and provides guidance to train the generator to generate undetectable fake user profiles. Each of the generators and the discriminator strikes to enhance

TABLE II
STATISTICS OF DATA

Data	#Users	#Items	#Ratings	Sparsity
ML-100K	943	1,682	100,000	93.70%
FilmTrust	780	721	28,799	94.88%
Yelp	2,762	10,477	119,237	99.59%
Automotive	2,928	1,835	20,473	99.62%
T & HI	1,208	8,491	28,396	99.72%
G & GF	2,212	8,041	62,424	99.65%
A & A	7.845	12,615	193,928	99.80%

TABLE III
ATTACK BUDGET

Data	A	P	|S|	#Targets
ML-100K	50	90	3	5
FilmTrust	50	36	3	5
Yelp	50	5	1	5
Automotive	50	4	1	5
T & HI	50	8	1	5
G & GF	50	10	1	5
A & A	50	9	1	5

itself to beat the other one at every round of the minimax competition.

VI. EXPERIMENT

In this section, we conduct experiments[3] to answer the following research questions.

1) *RQ1:* Does Leg-UP give better attack performance on different victim RS models, compared with other shilling attack methods? (Section VI-B).
2) *RQ2:* Is it more difficult to recognize the attack launched by Leg-UP, compared with other shilling attack methods? (Section VI-C).
3) *RQ3:* Does each component in Leg-UP contribute to its attack effects? (Section VI-D).
4) *RQ4:* Can Leg-UP achieve tailored attack goals such as In-segment Attacks? (Section VI-E).
5) *RQ5:* How does the attack budget affect the performance of attack? (Section VI-F).
6) *RQ6:* Does the choice of the surrogate model affect Leg-UP? (Section VI-G).

A. Experimental Setup

We use seven benchmark datasets for RS in our experiments, including ML-100K,[4] FilmTrust,[5] Yelp[6], and four other Amazon datasets[7] automotive, tools and home improvement (T & HI), Grocery and Gourmet Food (G & GF), and Apps for Android (A & A). ML-100K is used as the sole dataset in most previous work on shilling attacks [13] due to the convenience of computation. We use its default training/test split. However, we believe that the nature of datasets affects attack transferability. For example, some victim RS models perform better on denser datasets. Thus, we include the other four datasets, which are larger and sparser, to testify the competence of Leg-UP in different settings. We randomly split datasets except for ML-100K by 9:1 for training and testing, respectively. To exclude cold-start users (as they are too vulnerable), we filter users with less than 15 ratings and items without ratings. Five target items are randomly sampled for each dataset. Table II illustrates the statistics of the data, where "Sparsity" is the percentage of missing ratings.

The attack budget is shown in Table III. We use each attack method to generate 50 user profiles. This is roughly the population that can manifest the differences among attack models [9]. In each injected user profile, we demand the number of nonzero ratings to be equal to the average number of ratings per user in the dataset. This makes the "profile size" to be 90, 36, 5, 4, 8, 10, and 9 in ML-100K, FilmTrust, Yelp, Automotive, T & HI, G & GF, and A & A, respectively.

B. Attack Effectiveness (RQ1)

We compare Leg-UP with several shilling attack methods, including classic heuristic-based methods and recent GAN-based methods.

1) **AIA** stands for adversarial injection attack, a bilevel optimization framework to generate fake user profiles by maximizing the attack objective on the surrogate model [20]. AIA randomly selects A (i.e., the attack size) real user profiles and uses them to initialize the optimization. Following the original paper, we use WRMF as the surrogate model and unroll the last step as an approximation of the adversarial gradient. Note that AIA does not constrain the profile size. It is possible that a generated fake user profile has more than P filler items.
2) **DCGAN** is a GAN [43] adopted in a recent shilling attack method [18], where the generator takes noise and outputs fake user profiles through convolutional units. We follow the settings in [18] for the network structures and hyperparameters. We randomly sample P nonzero ratings in the fake user profiles for a fair comparison.
3) **WGAN** is the Wasserstein GAN [44] which has shown better empirical performance than the original GAN in many tasks [54]. We replace the GAN architecture in DCGAN with WGAN for shilling attacks. The hyperparameters are the same as DCGAN.
4) **Random Attack** is a heuristic-based method [9], which assigns random rating $\hat{\mathbf{X}}_{u,i} \sim \mathcal{N}(\mu, \sigma)$ to P random items in the fake user profile u, where P is the profile size, μ, and σ are the mean and the variance of all ratings in the system, respectively.
5) **Average Attack** [9] assumes that the fake user assigns a rating $\hat{\mathbf{X}}_{u,i} \sim \mathcal{N}(\mu_i, \sigma_i)$ to P randomly sampled items, where μ_i and σ_i are the mean and the variance of ratings on this item i, respectively.
6) **Segment attack** divides the fake user profile into selected items and filler items. It assigns maximal ratings to the selected items and minimal ratings to the filler items. Following [9], for each target item in ML-100K, we select three items that are most frequently rated under the same tag/category of the target item as the

[3]The source code of Leg-UP is available at https://github.com/XMUDM/ShillingAttack.
[4]https://grouplens.org/datasets/movielens/100k
[5]https://www.librec.net/datasets/filmtrust.zip
[6]https://www.kaggle.com/c/yelp-recruiting/data
[7]http://jmcauley.ucsd.edu/data/amazon

TABLE IV

ANALYSES OF ATTACK PERFORMANCE. (a) USE DEFAULT VALUES FOR A AND P. (b) USE VARIOUS VALUES FOR A AND P AS IN TABLE IX

(a)

Method	Leg-UP	AIA	DCGAN	WGAN	Random Attack	Average Attack	Segment Attack	Bandwagon Attack
#Best results	30	2	0	0	0	1	8	1
Ratio (Max: 100%)	71.43%	4.76%	0.00%	0.00%	0.00%	2.38%	19.05%	2.38%
#Top-2 results	40	8	0	0	1	3	28	4
Ratio (Max: 50%)	47.62%	9.52%	0.00%	0.00%	1.19%	3.57%	33.33%	4.76%

(b)

Method	Leg-UP	AIA	DCGAN	WGAN	Random Attack	Average Attack	Segment Attack	Bandwagon Attack
#Best results	147	5	1	0	8	6	40	3
Ratio (Max: 100%)	70.00%	2.38%	0.48%	0.00%	3.81%	2.86%	19.05%	1.43%
#Top-2 results	197	37	1	0	17	16	142	10
Ratio (Max: 50%)	46.91%	8.81%	0.24%	0.00%	4.05%	3.81%	33.81%	2.38%

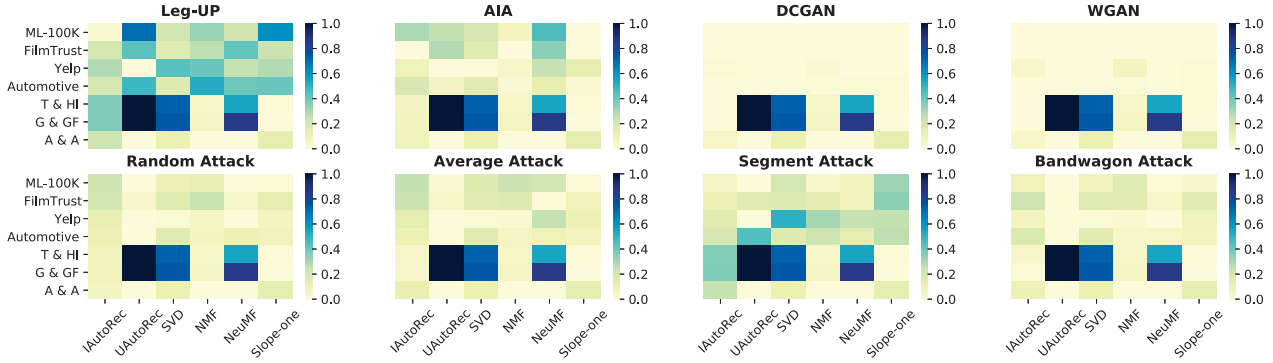

Fig. 4. Heatmaps showing attack results (default A and P). The darker a cell shows, the larger the corresponding HR@10 is.

selected items. For each target item in the other six datasets which do not have information on tag/category, we sample three items (FilmTrust) and one item (Yelp, Automotive, T & HI, G & GF, and A & A) from global popular items as the selected items [24].

7) **Bandwagon Attack** [21] uses the most popular items as the selected items and assigns the maximal rating to them, while fillers are assigned with ratings in the same manner as a random attack.

WGAN and DCGAN learn the rating assigned to the target item. In other attack methods for comparison, the highest possible rating (i.e., five-star) is assigned to the target item. The hidden layer in the AE (i.e., the generator) of Leg-UP [i.e., (2)] has 128 neurons. The first and the second hidden layers in the discriminator of Leg-UP [i.e., (13)] have 512 and 128 neurons, respectively. For AIA and Leg-UP, we use WRMF with $\lambda = 10^{-5}$ as the default surrogate model and optimize it with Adam [55].

We evaluate shilling attacks on a wide range of victim RS models before and after the attack, including shallow models [non-negative matrix factorization (NMF) [56], slope-one [57], and singular value decomposition (SVD)][8] and deep learning-based models (NeuMF [58] and variants of AutoEncoder [59], i.e., IAutoRec and UAutoRec).[9] Note that Leg-UP can generate discrete ratings. Since our goal is to estimate how the rating values in fake user profiles spoof RS models, we modify NeuMF, which is originally designed for implicit feedback, to predict explicit ratings, by using the RMSE loss.

Before attacking, we train each victim RS model on the training set. Then, we deploy each attack method to generate fake user profiles. The required information for each attack method (e.g., mean and variance) is obtained from the training set. After the injection, each victim's RS model will be trained again from scratch on the polluted data. We train victim RS models, attack competitors, and Leg-UP until convergence.

We evaluate the attack performance on the test set using Hit Ratios at K with $K = 10$ (i.e., HR@10) on the target item. The larger HR@10 is, the more effective the attacker is.

Table IV(a) illustrates how many times and ratios that different attackers produce the highest HR@10 (the maximum possible ratio is 100%) and the top-two highest HR@10 (the maximum possible ratio is 50% when an attacker produces top-two best results for all cases, and the highest HR@10 also counts) on different datasets using default values for A and P. And Fig. 4 provides heatmaps showing the attack performance of different attackers (using default A and P) on different victim RS models on various datasets. In Fig. 4, each cell indicates HR@10 of an attack method attacking a victim RS model on a dataset, and darker colors indicate higher HR@10 values. In the following, we provide analyses of the attack performance:

1) Overall Performance: From Table IV(a), we can conclude that, compared with other attack methods, Leg-UP generally achieves attractive attack performance against all victim RS models on different datasets. Using default settings of A and P, Leg-UP is consistently the best attack method compared with all competitors (71.43% of the time and the

[8]For all shallow models, we use implementations from https://github.com/NicolasHug/Surprise.

[9]For all deep learning-based models, we use implementations at https://github.com/cheungdaven/DeepRec.

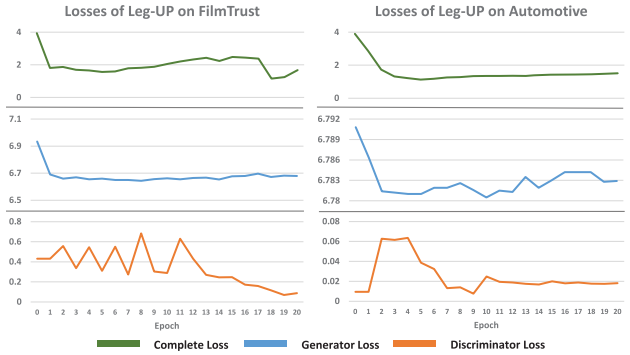

Fig. 5. Changes of losses in Leg-UP.

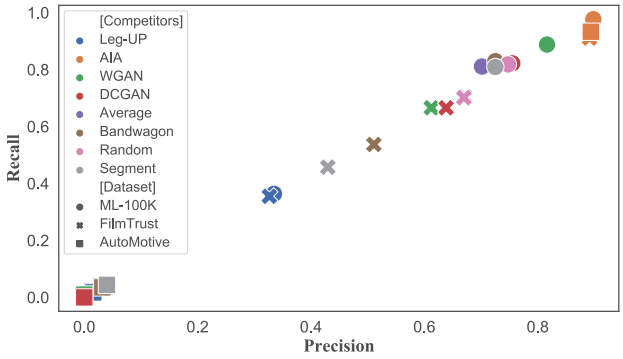

Fig. 6. Attack detection of injected profiles (precision versus recall). Low precision and recall values suggest an invisible attack model.

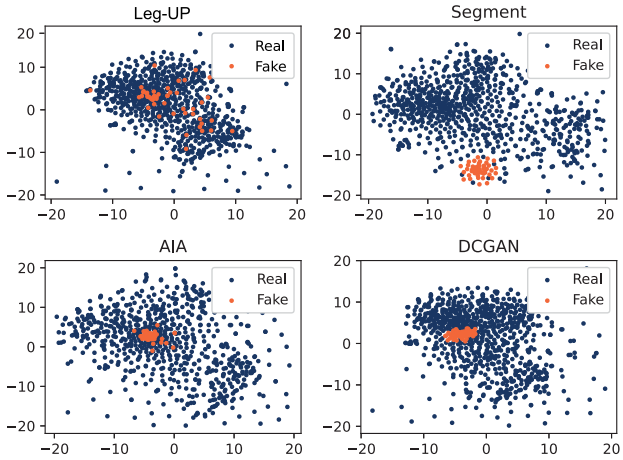

Fig. 7. Real and fake user profiles in the latent space.

maximum possible ratio is 100%), or the top-two best attack method (47.62% of the time and the maximum possible ratio is 50%). On the contrary, *conventional attack models do not show a robust attack performance like Leg-UP, even though they may exceed Leg-UP in a few cases.* This conclusion can be confirmed by observing Fig. 4: 1) Most cells in the heatmap of Leg-UP are dark (i.e., high HR@10), and they are darker than corresponding cells in other attackers' heatmaps. 2) Baselines' heatmaps have some cells in dark colors, but they all have a large region in a light color (i.e., low HR@10).

2) Comparisons Between Heuristic-Based and GAN-Based Methods: From Table IV(a), we can also find that heuristic-based methods are sometimes powerful. For example, segment attack is frequently the top-two best attack method (33.33% of the time), while *directly adopting the idea of adversarial attacks (i.e., using general GANs) does not give satisfactory performance*. Particularly, both DCGAN which is adopted in the recent shilling attacks [18] and WGAN [44] which aims at stabilizing GAN training do not show better performance than simple heuristic-based attack approaches like average attack and random attack. It shows that we need to tailor the idea of general GAN before applying it in shilling attacks.

3) Analysis of Different Losses in Leg-UP : Fig. 5 illustrates how the generator loss, the discriminator loss, and the complete loss [i.e., (15)] of Leg-UP (with default A and P) change on datasets FilmTrust and automotive during the optimization. As the generator and the discriminator compete with each other, we can see the fluctuation of their losses, and the increase of one loss is accompanied by the decrease of the other loss. On the whole, we can observe declining trends for both the generator loss and the discriminator loss, and therefore, the complete loss decreases during the optimization.

C. Attack Invisibility (RQ2)

1) Attack Detection: In order to testify how realistic the injected user profiles can be, we apply a state-of-the-art unsupervised attack detector [60] on the injected user profiles generated by different attack methods. Fig. 6 depicts the precision and recall rates of the detector on different attack methods. The lower values of precision and recall indicate that the attack method is more undetectable. We have the following observations based on the detection results

1) There is a positive correlation between precision and recall rates for all attack methods on different datasets.
2) The detection performance is highly dependent on the data and it is easy to detect fake profiles on denser datasets. For example, the detector struggles to detect fake profiles from almost all attackers on automotive. But it has high precision and recall rates for detecting most attack methods (except Leg-UP) on ML-100K.
3) Leg-UP consistently produces virtually undetectable injections. In most cases, the detector performs the worst against Leg-UP. On the contrary, the detection performance of baseline attackers is unstable.

2) Fake User Distribution: To further study the attack invisibility of Leg-UP, we visualize the real and fake user profiles using the t-SNE projection [61] of the latent space. Fig. 7 depicts the visualization results of Leg-UP, segment attack, AIA, and DCGAN. From the results, we can see that the fake user profiles produced by Leg-UP follow a similar distribution as real user profiles, i.e., fake users are scattered in the latent space of real users. As a comparison, fake users produced by other attackers collapse to a small region in the space, making them easy to detect. Therefore, we can conclude that Leg-UP preserves the diversity of real user behavior patterns in the RS (i.e., personalization), making it more undetectable than other shilling attack methods.

D. Ablation Study (RQ3)

To answer RQ3, we remove or change some components of Leg-UP and investigate the performance changes.

TABLE V

ABLATION STUDY: IMPACTS OF USING DIRECT GENERATION LOSS OR INDIRECT GENERATION LOSS ON ATTACK PERFORMANCE AND DETECTION PERFORMANCE. BETTER RESULTS ARE SHOWN IN BOLD. (a) ATTACK PERFORMANCE (HR@10). (b) DETECTION RESULTS

(a)

Dataset	ML-100K		FilmTrust		Automotive	
	Direct	Indirect	Direct	Indirect	Direct	Indirect
IAutoRec	0.0145	**0.0990**	0.7180	**0.8333**	**0.9958**	0.9955
UAutoRec	**0.6450**	0.2006	**0.9997**	0.3549	0.6369	**0.8000**
SVD	**0.8575**	0.5521	**0.9588**	0.9245	**0.8776**	0.8649
NMF	**0.2461**	0.1849	**0.3443**	0.2821	**0.1642**	0.1466
NeuMF	0.1833	**0.3131**	0.4546	**0.6868**	0.3333	**0.3712**
Slope-one	**0.5835**	0.0258	**0.4061**	0.0678	**0.1133**	0.0869

(b)

Dataset	ML-100K		FilmTrust		Automotive	
	Direct	Indirect	Direct	Indirect	Direct	Indirect
Precision	0.3347	**0.1522**	0.3265	**0.3107**	**0.0160**	0.0600
Recall	0.3638	**0.1830**	0.3552	**0.3238**	**0.0176**	0.0660

TABLE VI

ABLATION STUDY: COMPARISONS BETWEEN USING AUTOENCODER (AE) OR USING A SIMPLE FUNCTION (SF AS THE GENERATOR IN LEG-UP. BETTER RESULTS ARE SHOWN IN BOLD. (a) ATTACK PERFORMANCE. (b) DETECTION RESULTS

(a)

Dataset	ML-100K		FilmTrust		Automotive	
	AE	SF	AE	SF	AE	SF
IAutoRec	0.0145	**0.0448**	0.7180	**0.8229**	**0.9955**	0.9677
UAutoRec	**0.6450**	0.3644	**0.9997**	0.2561	0.6369	**0.7886**
SVD	**0.8575**	0.5095	**0.9588**	0.9430	**0.8776**	0.8580
NMF	**0.2461**	0.1364	**0.3443**	0.1397	**0.1642**	0.1030
NeuMF	0.1833	**0.2793**	0.4546	**0.5631**	0.3333	**0.2320**
Slope-one	**0.5835**	0.0094	**0.4061**	0.0631	**0.1133**	0.0731

(b)

Dataset	ML-100K		FilmTrust		Automotive	
	AE	SF	AE	SF	AE	SF
Precision	0.3347	**0.2974**	0.3265	**0.1845**	0.0160	**0.0000**
Recall	0.3638	**0.3240**	0.3552	**0.2000**	0.0176	**0.0000**

1) Impacts of Generation Loss: As mentioned in Section IV, a direct "attack-related" loss [(11)] is beneficial to the improvement of the attack performance, while an indirect "reconstruction-related" loss [(7)] can improve attack invisibility. This is supported by Table V, which reports the attack performance and detection results of Leg-UP (i.e., direct loss), where the generation loss is directly measured upon the attack performance, and AUSH (i.e., indirect loss), where the generation loss is measured upon the reconstruction performance. We can see that, in most cases, the indirect loss is outperformed by direct loss in terms of HR@10. However, with indirect loss, AUSH can more often generate indistinguishable fake user profiles than Leg-UP.

2) Impacts of Generator Architecture: Leg-UP can use a simple function or a complex neural network module as the generator. Note that, when using the simple function, Leg-UP resembles AIA, where ratings of "template" users are used to initiate the optimization process. When using a neural network module like an autoencoder, the parameters of the neural network instead of the rating values themselves will be updated during optimization. As shown in Table VI, removing the autoencoder decreases HR@10. Adopting a nonlinear module, such as an autoencoder, can capture the complex, or even high-order interactions among users and items. As a result, the generated fake user profiles can sabotage the victim RS models and the attack transferability is improved.

TABLE VII

ABLATION STUDY: COMPARISONS BETWEEN USING DISCRETIZATION WITH A LEARNABLE LAYER (DL) OR USING SR. BETTER RESULTS ARE SHOWN IN BOLD. (a) ATTACK PERFORMANCE. (b) DETECTION RESULTS

(a)

Dataset	ML-100K		FilmTrust		Automotive	
	DL	SR	DL	SR	DL	SR
IAutoRec	**0.0145**	0.0049	**0.7180**	0.5379	**0.9955**	0.8858
UAutoRec	**0.6450**	0.4304	**0.9997**	0.9869	**0.6369**	0.6000
SVD	**0.8575**	0.6465	0.9588	**0.9632**	**0.8776**	0.8392
NMF	**0.2461**	0.0082	0.3443	**0.3471**	**0.1642**	0.0790
NeuMF	**0.1833**	0.0162	**0.4546**	0.2450	**0.3333**	0.2611
Slope-one	**0.5835**	0.5165	0.4061	**0.4198**	**0.1133**	0.0441

(b)

Dataset	ML-100K		FilmTrust		Automotive	
	DL	SR	DL	SR	DL	SR
Precision	0.3347	**0.1440**	0.3265	**0.1959**	**0.0160**	0.0360
Recall	0.3638	**0.1596**	0.3552	**0.2130**	**0.0176**	0.0396

TABLE VIII

ABLATION STUDY: COMPARISONS BETWEEN LEG-UP WITH (W. D) AND WITHOUT (W/O. D) THE DISCRIMINATOR. BETTER RESULTS ARE SHOWN IN BOLD. (a) ATTACK PERFORMANCE. (b) DETECTION PERFORMANCE

(a)

Dataset	ML-100K		FilmTrust		Automotive	
	w. D	w/o. D	w. D	w/o. D	w. D	w/o. D
IAutoRec	0.0145	**0.0874**	**0.7180**	0.5663	**0.9955**	0.9565
UAutoRec	0.6450	**0.8364**	0.9997	**1.0000**	**0.6369**	0.0003
SVD	0.8575	**0.9323**	**0.9588**	0.9582	**0.8776**	0.8297
NMF	0.2461	**0.2771**	0.3443	**0.3580**	**0.1642**	0.0923
NeuMF	0.1833	**0.5627**	0.4546	**0.9293**	**0.3333**	0.2807
Slope-one	0.5835	**0.6292**	**0.4061**	0.4059	**0.1133**	0.0459

(b)

Dataset	ML-100K		FilmTrust		Automotive	
	w. D	w/o. D	w. D	w/o. D	w. D	w/o. D
Precision	**0.3347**	0.4204	**0.3265**	0.4250	**0.0160**	0.0200
Recall	**0.3638**	0.4572	**0.3552**	0.4528	**0.0176**	0.0200

TABLE IX

PERFORMANCE OF LEG-UP WITH VARYING A AND P

ML-100K	**A**	50	50	50	38	64
	P	75	90	110	90	90
	HR@10	0.1550	0.1833	0.1916	0.1501	0.2644
FilmTrust	**A**	50	50	50	25	75
	P	22	36	50	36	36
	HR@10	0.3639	0.4546	0.1821	0.1035	0.6157
Automotive	**A**	50	50	50	38	64
	P	2	4	8	4	4
	HR@10	0.3224	0.3333	0.4338	0.3847	0.3224
T & HI	**A**	50	50	50	38	64
	P	5	8	15	8	8
	HR@10	0.6205	0.5488	0.6161	0.5620	0.5230
G & GF	**A**	50	50	50	38	64
	P	5	10	15	10	10
	HR@10	0.8552	0.8561	0.8522	0.8813	0.8715

3) Impacts of Discretization Strategy: Since typical RS uses Likert-scale ratings, it is necessary to generate fake user profiles with discrete ratings (e.g., one to five stars) to mimic the normal interactions between normal users and RS. Leg-UP provides two strategies: simply using rounding to map numerical values (i.e., user preferences) to discrete stars like

TABLE X
PERFORMANCE OF LEG-UP AND AIA USING DIFFERENT SURROGATE MODELS WITH DEFAULT A AND P

Dataset	Attacker	Surrogate Model	IAutoRec	UAutoRec	SVD	NMF	NeuMF	Slope-one
T & HI	Leg-UP	IAutoRec	0.7083	0.0041	0.7167	0.0573	0.0157	0.2301
		SVD++	0.9348	1.0000	0.7336	0.0616	0.0169	0.4162
	AIA	IAutoRec	0.1858	0.0000	0.7052	0.0067	0.0056	0.1481
		SVD++	0.1998	0.0000	0.7125	0.0070	0.0062	0.1596
G & GF	Leg-UP	IAutoRec	0.8664	0.7880	0.7364	0.0615	0.0065	0.7648
		SVD++	0.8694	1.0000	0.7401	0.0699	0.0072	0.8264
	AIA	IAutoRec	0.2009	0.0007	0.7413	0.0035	0.0028	0.1537
		SVD++	0.2084	0.0006	0.7400	0.0035	0.0031	0.1474

AUSH [24], or learning how to discretize user preferences into discrete ratings. In Table VII, we report results produced by the DL or the SR. We can see that discretization with learnable thresholds helps raise HR@10 because it involves the discretization in the optimization of the overall attack framework so that the discretization errors can be reduced during optimization.

4) Impacts of Using a Discriminator: Leg-UP is built upon the idea of GAN, where the discriminator is incorporated to guide the generator to produce "perfect" fake user profiles. To study the impacts of using the discriminator, we report HR@10 by only using the generator in Leg-UP in Table VIII. We can see that incorporating the discriminator is usually harmful in terms of HR@10. However, using the discriminator is indispensable to generate undetectable fake user profiles for Leg-UP. Therefore, we consider it as a flexible option to include discriminator in Leg-UP to launch an effective attack which should be difficult for detectors to discover.

E. In-Segment Attack Goals (RQ4)

Like some existing shilling attack methods [18], [24], it is easy to modify Leg-UP to achieve a tailored attack goal, e.g., attacking in-segment users as a segment attack does. In-segment users [12] are users who have shown preferences on selected items. They are a popular target audience because companies which face fierce competition are eager to attract the market population of their competitors. We define in-segment users as users who have assigned high ratings (i.e., four or five stars) on all selected item in the training set. We can modify the generation loss in (8) so that in-segment attack can be launched by Leg-UP

$$\mathcal{L}_{\text{gen}} = -\sum_{u \in \mathcal{U}^{(\text{seg})}} \log \frac{\exp \tilde{\mathbf{X}}_{u,t}}{\sum_{j \in \mathcal{I}} \exp \tilde{\mathbf{X}}_{u,j}} \tag{18}$$

where $\mathcal{U}^{(\text{seg})}$ is the set of in-segment users and t is the target item.

Fig. 8 reports HR@10 of segment attack, Leg-UP and Leg-UP with the tailored loss [i.e., Leg-UP (seg)] against different victim RS models on in-segment users for the automotive dataset. We can observe that by modifying the objective, Leg-UP can always obtain higher HR@10 results on in-segment users. We can also find that Leg-UP generally has better attack performance on in-segment users than on segment attacks.

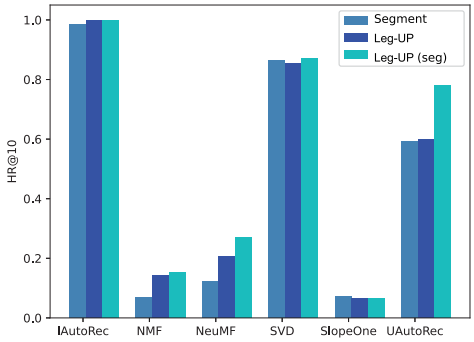

Fig. 8. HR@10 on in-segment users for Automotive. Higher value suggests a better attack method.

F. Effects of Attack Budget (RQ5)

We also investigate the effect of the attack budget hyperparameter A and P on Leg-UP. In Table IX, we report the performance of Leg-UP using varying A and P. We can observe that larger values of A improve the attack performance of Leg-UP as more fake user profiles are injected to spoof the system. For the profile size P, Leg-UP achieves reasonable performance using the default values (i.e., the average number of ratings per user in the dataset) on different datasets, justifying the default setting for p. Compared with default values of P, smaller or larger P values reduce the attack performance of Leg-UP or only slightly improve the attack performance of Leg-UP.

Moreover, we compare Leg-UP to other attack methods using varying A and P (the values are listed in Table IX) on different datasets. Table IV(b) reports how many times and ratios that Leg-UP produces the highest HR@10 and the top-two highest HR@10. Similar to the results depicted in Table IX, Table IV(b) shows that Leg-UP consistently achieves the best attack performance most of the time when different A and P are used in the experiments. Hence, we can conclude that Leg-UP is robust regardless of the setting of the attack budget.

G. Effects of Surrogate Models (RQ6)

By default, we adopt WRMF as the surrogate model in Leg-UP and AIA. We further investigate the effects of surrogate models by using another two RS models IAutoRec and SVD++ [62]. Table X reports the attack performance of Leg-UP and AIA on two datasets T & HI and G & GF, using default settings for the attack budget A and P. From the results in Table X and Fig. 4, we can observe that different

surrogate models affect the attack performance. However, most of the time, changing the surrogate model in one attack method does not affect the performance too much. Moreover, the attack performance of Leg-UP is robust, i.e., it consistently outperforms AIA by a large margin regardless of the adopted surrogate model.

VII. Conclusion

A shilling attack is one of the most subsistent and profitable attack types against RS. By injecting a small number of fake user profiles, where each profile contains ratings on some items, the target items receive more (or less) recommendations by the victim RS model. We show that two challenges arise in designing shilling attack methods: low attack transferability and low attack invisibility. To overcome these challenges, in this article, we present a novel shilling attack framework Leg-UP based on the idea of GAN. We discuss and provide different options for the design of the generator module, the generation loss, the discriminator module, and the learning method for Leg-UP. The experimental results show the superiority of Leg-UP over state-of-the-art shilling attack methods. Leg-UP is effective against a wide range of shallow and deep RS models on several benchmark RS datasets. Meanwhile, Leg-UP is virtually undetectable by the modern RS attack detector. Leg-UP, as a new shilling attack method, can benefit the study of secure and robust RS.

In the future, an interesting research direction is to study the impacts of surrogate models on attack transferability. We also plan to study the learning process of Leg-UP for more efficient learning. Furthermore, user feedback in RS is usually sequential or multimodal. Therefore, extending Leg-UP to sequential or multimodal data is attractive.

References

[1] C. C. Aggarwal, *Recommender Systems—The Textbook*. Cham, Switzerland: Springer, 2016.

[2] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 3:1–3:45, 2014.

[3] Y. Deldjoo, T. Di Noia, and F. A. Merra, "A survey on adversarial recommender systems: From attack/defense strategies to generative adversarial networks," 2020, *arXiv:2005.10322*.

[4] M. Pang, W. Gao, M. Tao, and Z. Zhou, "Unorganized malicious attacks detection," in *Proc. NIPS*, 2018, pp. 6976–6985.

[5] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov, "'You might also like': Privacy risks of collaborative filtering," in *Proc. IEEE Symp. Secur. Privacy*, May 2011, pp. 231–246.

[6] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: A comprehensive survey," *Artif. Intell. Rev.*, vol. 42, no. 4, pp. 767–799, Dec. 2014.

[7] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Proc. NIPS*, 2016, pp. 1885–1893.

[8] H. Chen and J. Li, "Data poisoning attacks on cross-domain recommendation," in *Proc. CIKM*, 2019, pp. 2177–2180.

[9] R. D. Burke, B. Mobasher, R. Bhaumik, and C. Williams, "Segment-based injection attacks against collaborative filtering recommender systems," in *Proc. ICDM*, 2005, pp. 577–580.

[10] X. Xing, W. Meng, D. Doozan, A. C. Snoeren, N. Feamster, and W. Lee, "Take this personally: Pollution attacks on personalized services," in *Proc. USENIX Secur. Symp.*, 2013, pp. 671–686.

[11] G. Yang, N. Z. Gong, and Y. Cai, "Fake co-visitation injection attacks to recommender systems," in *Proc. NDSS*, 2017, pp. 1–15.

[12] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proc. WWW*, 2004, pp. 393–402.

[13] J. J. Sandvig, B. Mobasher, and R. D. Burke, "A survey of collaborative recommendation and the robustness of model-based algorithms," *IEEE Data Eng. Bull.*, vol. 31, no. 2, pp. 3–13, Jun. 2008.

[14] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

[15] J. Su, D. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.

[16] M. Alzantot, Y. Sharma, A. Elgohary, B. Ho, M. B. Srivastava, and K. Chang, "Generating natural language adversarial examples," in *Proc. EMNLP*, 2018, pp. 2890–2896.

[17] K. Christakopoulou and A. Banerjee, "Adversarial recommendation: Attack of the learned fake users," 2018, *arXiv:1809.08336*.

[18] K. Christakopoulou and A. Banerjee, "Adversarial attacks on an oblivious recommender," in *Proc. RecSys*, 2019, pp. 322–330.

[19] J. Song *et al.*, "Poisonrec: An adaptive data poisoning framework for attacking black-box recommender systems," in *Proc. ICDE*, 2020, pp. 157–168.

[20] J. Tang, H. Wen, and K. Wang, "Revisiting adversarially learned injection attacks against recommender systems," in *Proc. RecSys*, 2020, pp. 318–327.

[21] R. Burke, B. Mobasher, and R. Bhaumik, "Limited knowledge shilling attacks in collaborative filtering systems," in *Proc. ITWP@IJCAI*, 2005, pp. 17–24.

[22] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Trans. Internet Technol.*, vol. 7, no. 3, p. 23, Oct. 2007.

[23] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 5:1–5:38, 2019.

[24] C. Lin, S. Chen, H. Li, Y. Xiao, L. Li, and Q. Yang, "Attacking recommender systems with augmented user profiles," in *Proc. CIKM*, 2020, pp. 855–864.

[25] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.

[26] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, "A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation," 2021, *arXiv:2104.13030.*

[27] T. X. Tuan and T. M. Phuong, "3D convolutional networks for session-based recommendation with content features," in *Proc. RecSys*, 2017, pp. 138–146.

[28] T. Moins, D. Aloise, and S. J. Blanchard, "RecSeats: A hybrid convolutional neural network choice model for seat recommendations at reserved seating venues," in *Proc. RecSys*, 2020, pp. 309–317.

[29] K. Zhou, H. Yu, W. X. Zhao, and J.-R. Wen, "Filter-enhanced MLP is all you need for sequential recommendation," 2022, *arXiv:2202.13556*.

[30] X. Zhang, Z. Wang, and B. Du, "Deep dynamic interest learning with session local and global consistency for click-through rate predictions," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3306–3315, May 2022.

[31] H. Xia, Z. Wang, B. Du, L. Zhang, S. Chen, and G. Chun, "Leveraging ratings and reviews with gating mechanism for recommendation," in *Proc. CIKM*, 2019, pp. 1573–1582.

[32] T. Huang *et al.*, "MixGCF: An improved training method for graph neural network-based recommender systems," in *Proc. KDD*, 2021, pp. 665–674.

[33] C. Huang *et al.*, "Knowledge-aware coupled graph neural network for social recommendation," in *Proc. AAAI*, 2021, pp. 4115–4122.

[34] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 3, pp. 1–41, 2020.

[35] N. N. Dalvi, P. M. Domingos, Mausam, S. K. Sanghai, and D. Verma, "Adversarial classification," in *Proc. KDD*, 2004, pp. 99–108.

[36] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," in *Proc. ECML/PKDD*, vol. 8190, 2013, pp. 387–402.

[37] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proc. AsiaCCS*, 2006, pp. 16–25.

[38] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, 2010.

[39] F. Roli, B. Biggio, and G. Fumera, "Pattern recognition systems under attack," in *Proc. CIARP*, vol. 8258, 2013, pp. 1–8.

[40] Y. Hong, U. Hwang, J. Yoo, and S. Yoon, "How generative adversarial networks and their variants work: An overview," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 10:1–10:43, 2019.

[41] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. AAAI*, 2017, pp. 2852–2858.
[42] C. Wang, M. Niepert, and H. Li, "RecSys-DAN: Discriminative adversarial networks for cross-domain recommender systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2731–2740, Aug. 2020.
[43] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.
[44] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, *arXiv:1701.07875*.
[45] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," 2018, *arXiv:1710.11342*.
[46] C. Xiao, B. Li, J. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," in *Proc. IJCAI*, 2018, pp. 3905–3911.
[47] M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre, "Recommender systems: Attack types and strategies," in *Proc. AAAI*, 2005, pp. 334–339.
[48] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, "Collaborative recommendation: A robustness analysis," *ACM Trans. Internet Technol.*, vol. 4, no. 4, pp. 344–377, 2004.
[49] D. C. Wilson and C. E. Seminario, "When power users attack: Assessing impacts in collaborative recommender systems," in *Proc. RecSys*, 2013, pp. 427–430.
[50] C. E. Seminario and D. C. Wilson, "Attacking item-based recommender systems with power items," in *Proc. RecSys*, 2014, pp. 57–64.
[51] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in *Proc. ACSAC*, 2018, pp. 381–392.
[52] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," in *Proc. AAAI*, 2015, pp. 2871–2877.
[53] N. Tsoi, Y. Milkessa, and M. Vázquez, "A heaviside function approximation for neural network binary classification," 2020, *arXiv:2009.01367*.
[54] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. NIPS*, 2017, pp. 5767–5777.
[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015, *arXiv:1412.6980*.
[56] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. NIPS*, 2000, pp. 556–562.
[57] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," in *Proc. SDM*, 2005, pp. 471–475.
[58] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. WWW*, 2017, pp. 173–182.
[59] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proc. WWW Companion*, 2015, pp. 111–112.
[60] Y. Zhang, Y. Tan, M. Zhang, Y. Liu, T. Chua, and S. Ma, "Catch the black sheep: Unified framework for shilling attack detection based on fraudulent action propagation," in *Proc. IJCAI*, 2015, pp. 2408–2414.
[61] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[62] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. KDD*, 2008, pp. 426–434.

Chen Lin (Member, IEEE) received the B.Eng. and Ph.D. degrees from Fudan University, Shanghai, China, in 2004 and 2010, respectively.

She is currently a Professor with the School of Informatics, Xiamen University, Xiamen, China. Her research interests include Web mining and recommender systems.

Si Chen received the master's degree from the School of Informatics, Xiamen University, Xiamen, China, in 2021.

Her research interests include data mining, recommender systems, and computational advertising.

Meifang Zeng is currently pursuing the master's degree with the School of Informatics, Xiamen University, Xiamen, China.

Her research interests include data mining and recommender systems.

Sheng Zhang is currently pursuing the master's degree with the School of Informatics, Xiamen University, Xiamen, China.

His research interests include data mining and recommender systems.

Min Gao (Member, IEEE) received the M.S. and Ph.D. degrees in computer science from Chongqing University, Chongqing, China, in 2005 and 2010, respectively.

She is an Associate Professor with the School of Big Data and Software Engineering, Chongqing University. Her research interests include recommender systems, shilling attack detection, and service computing.

Hui Li (Member, IEEE) received the B.Eng. degree in software engineering from Central South University, Changsha, China, in 2012, and the M.Phil. and Ph.D. degrees in computer science from The University of Hong Kong, Hong Kong, in 2015 and 2018, respectively.

He is currently an Assistant Professor with the School of Informatics, Xiamen University, Xiamen, China. His research interests include data mining and data management with applications in recommender systems and knowledge graphs.