Computers, Materials & Continua
DOI:10.32604/cmc.2022.020437
Article

Generating A New Shilling Attack for Recommendation Systems

Pradeep Kumar Singh[1], Pijush Kanti Dutta Pramanik[1], Madhumita Sardar[1], Anand Nayyar[2,3,*], Mehedi Masud[4] and Prasenjit Choudhury[1]

[1]Department of Computer Science & Engineering, National Institute of Technology, Durgapur, India
[2]Graduate School, Duy Tan University, Da Nang, Vietnam
[3]Faculty of Information Technology, Duy Tan University, Da Nang, Vietnam
[4]Department of Computer Science, College of Computers and Information Technology, Taif University, Taif, 21944, Saudi Arabia
*Corresponding Author: Anand Nayyar. Email: anandnayyar@duytan.edu.vn
Received: 24 May 2021; Accepted: 27 June 2021

Abstract: A collaborative filtering-based recommendation system has been an integral part of e-commerce and e-servicing. To keep the recommendation systems reliable, authentic, and superior, the security of these systems is very crucial. Though the existing shilling attack detection methods in collaborative filtering are able to detect the standard attacks, in this paper, we prove that they fail to detect a new or unknown attack. We develop a new attack model, named Obscure attack, with unknown features and observed that it has been successful in biasing the overall top-N list of the target users as intended. The Obscure attack is able to push target items to the top-N list as well as remove the actual rated items from the list. Our proposed attack is more effective at a smaller number of k in top-k similar user as compared to other existing attacks. The effectivity of the proposed attack model is tested on the MovieLens dataset, where various classifiers like SVM, J48, random forest, and naïve Bayes are utilized.

Keywords: Shilling attack; recommendation system; collaborative filtering; top-N recommendation; biasing; shuffling; hit ratio

1 Introduction

Recommendation systems are used by e-commerce companies to provide the best services to their customers by recommending items that they would like [1]. Appropriate recommendations have become a challenging task in the presence of overwhelmed data. In such a scenario, the recommendation systems employ a variety of filtering techniques to retrieve useful information from a huge amount of data. A recommendation system identifies a set of n specific items which are supposed to be the most appealing to the user and is termed as the top-N recommendation for that user [2–4]. Collaborative filtering is frequently used for recommending top-N items by considering the preferences of top-k similar users to the target user. User-based collaborative filtering techniques adopt similarity calculation to find the top-k similar users, and then the ratings given by these top-k similar users are utilized in the rating prediction for an unrated item [5]. In item-based collaborative filtering techniques, several item-based top-N recommendation algorithms are proposed, which utilize the rating information of users and similarity values between items [6]. The overall top-N list assures the most suitable products as it includes the best items across all the items in the system. However, the generated top-N lists by the predicted rating may be vulnerable in the presence of shilling attackers. A shilling attack involves inserting fake user profiles into a database to change the recommended top-N list of items [7]. The general objective of the attacker is to bias the overall top-N list as well as the top-N recommendation of a user.

1.1 Effects of Shilling Attacks in Collaborative Filtering Based Recommendation Systems

Attackers may change the list of the target users' nearest neighbors. Since, in collaborative filtering based recommendation system, users get item recommendation that very much depends on their nearest neighbors. Therefore, if the neighbor list is changed by a shilling attack, the recommendation is also affected accordingly. Due to this, the non-desirable items may be recommended to the users, and the loyalty of the recommendation system may decrease significantly. To elaborate on the effect of shilling attacks in the recommendation system, let us consider the following two scenarios.

Scenario 1: Let us assume there are four users and their rating vectors for ten items are as shown in Tab. 1. Here, 0 denotes that a user did not rate the particular item, the sets of Authentic users = $\{U_1, U_2, U_3, U_4, U_5, U_6\}$, Target users = $\{U_1, U_2, U_3, U_4\}$, and Attackers = $\{U_7, U_8, U_9, U_{10}\}$. We applied the existing shilling attacks (discussed in detail in Section 2.1) on the considered dataset of Tab. 1. From Tab. 2, we observe that the Random, Average, and Bandwagon attacks have changed the top-3 nearest neighbors of the user U_3 whereas in case of the Obfuscated attack, all users' neighbors are changed except U_2.

Table 1: User-item rating dataset

User	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
U_1	0	0	5	1	5	4	4	3	2	4
U_2	0	0	4	0	4	3	0	4	0	0
U_3	0	0	0	2	4	5	5	0	0	3
U_4	0	0	4	0	3	4	4	2	0	4
U_5	5	4	5	1	4	3	0	2	1	0
U_6	4	4	4	2	4	0	0	0	2	4

Table 2: Top-3 similar neighbors of the target users before and after various shilling attacks

User	Authentic	Random attack	Average attack	Bandwagon attack	Segment attack	Obfuscated attack
U_1	U_4, U_2, U_3	U_4, U_2, U_3	U_4, U_2, U_3	U_4, U_2, U_3	U_4, U_2, U_3	U_4, U_9, U_7
U_2	U_1, U_4, U_5	U_1, U_4, U_5	U_1, U_4, U_5	U_1, U_4, U_5	U_1, U_4, U_5	U_1, U_4, U_5
U_3	U_4, U_1, U_2	U_7, U_8, U_{10}	U_9, U_7, U_8	U_9, U_7, U_8	U_4, U_1, U_2	U_4, U_8, U_9
U_4	U_1, U_3, U_2	U_1, U_3, U_2	U_1, U_3, U_2	U_1, U_3, U_2	U_1, U_3, U_2	U_1, U_3, U_7

Scenario 2: Suppose we want to change the authentic item recommendation list of the target users. For example, let's say, from Tab. 3, the attackers aim to remove I_{10}, insert I_9, and change the preference of I_7 in the recommendation list of target users. Tab. 3 shows that the recommended item lists of the target users after shilling attacks. Here, we considered those items that had a predicted rating greater than or equal to 3.

Table 3: Recommended items to the target users before and after shilling attacks

User	Authentic	Random attack	Average attack	Bandwagon attack	Segment attack	Obfuscated attack
U_1	-	-	-	-	-	-
U_2	I_7, I_{10}	I_7, I_{10}	I_7, I_{10}	I_7, I_{10}	I_7, I_{10}	I_7, I_{10}
U_3	I_3, I_8	I_1, I_2	I_1, I_2	I_1, I_2, I_9	I_3, I_8	I_3
U_4	-	-	-	-	-	-

1.2 Purpose of Developing a New Shilling Attack

From Tab. 2, it can be concluded that though the existing shilling attacks can change the target users' nearest neighbors to some extent they are not successful in every case. Here, the Obfuscated attack, proposed by Chad et al. [8], seems to have the best performance, though not optimal. Similarly, from Tabs. 2 and 3, we can note that all other shilling attacks achieve the target to some extent, except Segment attack, but no attacks fulfill all the targets in the computed scenario. This opens up the scope for a new attack which can disrupt the list of nearest neighbors of any target user to the intended extent.

1.3 Implications of a New Shilling Attack

Attacking a system is not always ill-purposed. Attacks may be generated to test and make the system more robust and attack-proof. The positive implications and take away of a shilling attack on a recommendation system include:

- Although there are numerous works in the field of collaborative filtering based recommendation systems to detect shilling attacks, existing feature-based detection schemes may fail in the case of unknown attacks. An authorized person can use a new attack model as ethical hacking to reveal vulnerabilities in the collaborative filtering based recommendation system.
- A new shilling attack model may suggest how to protect the recommendation system, what are the attack entry points, and what can be done to improve the performance of the recommendation system from this type of attack.
- If a suitable item that meets the quality of Top-N recommended items is initially poorly rated or not rated at all by the users, then over time, it would eventually be removed from the recommended list. In this case, good or niche products may invisible to the consumers as they have never been reviewed or did not get a chance to get into the Top-N list for recommendation. A new shilling attack model may be used to mitigate the aforesaid condition, i.e., a long-tail problem.

Therefore, despite the fact that the term "attack" often refers to a negative operation, a new shilling attack can be used to increase the consistency and quality of the recommendation systems.

1.4 Contribution of This Paper

In this paper, we generate a new attack model with unknown features and prove the failure of the known feature-based detection techniques in identifying them. The major contributions of this paper are as below:

- Design and generate a new shilling attack for recommender systems.
- Compare the biasing, shuffling, and hit ratio of different attacks on the top-N list to find the attack size.
- Apply feature-based attack detection schemes to the known and proposed attacks on the calculated attack size and compare the results.

1.5 Organisation of the Paper

The requisite theoretical backgrounds that include the basic terms related to shilling attacks, attributes/features of attackers, and the metrics for assessing the effectiveness of shilling attacks, are discussed in Section 2. Section 3 includes the related work. Section 4 represents the proposed attack model with its validation. In Section 5, an experimental analysis of the effect of the proposed attack on the recommendation system is discussed. And finally, the conclusion and future work is presented in Section 6.

2 Theoretical Background

The basic structure of the collaborative filtering based recommendation system is depicted in Fig. 1. Data Collection, rating prediction using computed similarity, and top-N recommendation are the key components of this framework [9].

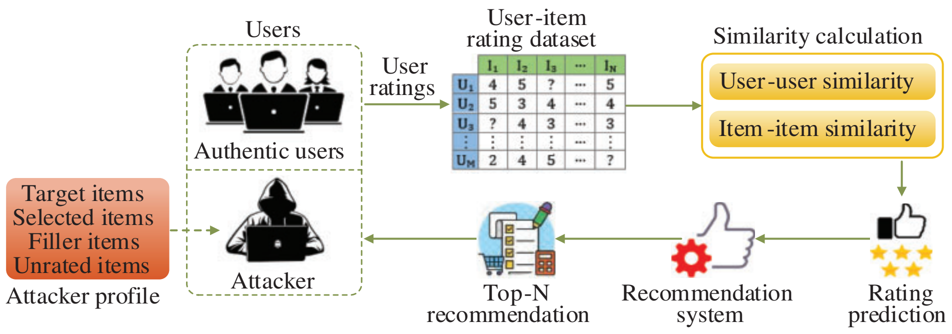

Figure 1: Conceptual framework of collaborative filtering

2.1 Basic Terms Related to Shilling Attacks

An attack is launched by creating a number of fake profiles. Based on the intent, an attack could either be a push or a nuke [10]. The target items are accordingly rated higher or lower than their average rating in order to have a strong impact on the weighted sum [11]. The target item is usually assigned the highest (for a push attack) or the lowest rating (for a nuke attack). The attackers must mimic the behavior of the authentic users to look normal and become the nearest neighbors of the target users. The attacker's profiles have to be designed in such a way that meets the above-mentioned requirements.

2.1.1 Components of an Attacker's Profile

Every attacker's profile is comprised of the following four parts [12]:

Target items (I_T): A singleton item whose recommendation is to be biased and is rated abnormally high (push) or low (nuke).

Selected items (I_S): A set of items whose rating is determined by a function based on the type of attack.

Filler items (I_F): A set of items randomly picked up and assigned random ratings so that the attacker apparently looks like a genuine user.

Unrated items (I_N): A set of unrated items.

2.1.2 Standard Shilling Attacks

Different attack models exist due to the differences in the rating functions and selection of I_F and I_S. The following attacks are considered as the standard attack models in the literature [13–15].

Random attack: $I_S = \emptyset$ and $\rho(i) = \mathrm{N}(\bar{r},\ \sigma^2)$, Lam and Riedl [16] introduced this attack. Here, I_S is empty, the ratings of I_F are selected randomly, and the function $\mathrm{N}(\mathrm{r},\sigma^2)$ produces random ratings centered on the overall average rating in the rating matrix. Here, σ^2 represents the variance and $\bar{r}$ denotes the average rating over all items and users. $\mathrm{N}(\mathrm{r},\sigma^2)$ denotes the Gaussian distribution, and $\rho(i)$ is the function that determines the ratings of the I_F.

Average attack: In the average attack $I_S = \emptyset$ and $\rho(\mathrm{i})) = \mathrm{N}(\overline{r_i}\ ,\ \sigma_i^2)$. The average attack is very much similar to random attack except that the ratings for I_F are generated by a function of $\mathrm{N}(\overline{r_i}\ ,\ \sigma_i^2)$, and centered around the average rating of each item in the database [16]. Here, σ_i is the standard deviation of ratings of item i over all those users who have rated this item and $\overline{r_i}$ is the average rating of items i over all the users who have rated this item.

Bandwagon attack: I_S contains some popular items which are assigned high ratings and $\rho(i) = \mathrm{N}(\overline{r_i}\ ,\ \sigma_i^2)$. The bandwagon attack is quite similar to the random attack, except that it needs to identify a set of most popular items in a particular domain.

Segment attack: I_S contains items similar to target items and $\rho(i) = \mathrm{N}(\overline{r_i}\ ,\ \sigma_i^2)$. The attack aims to increase the similarity between the target item and the items in I_S. The I_S comprises of those items which are liked by the target set of users [17]. They are rated with high values so that they become similar to the target items, as the target items are also given high ratings in a push attack. On the other hand, the I_F are assigned low ratings to make them different from the target item.

2.2 Attacker's Features

The attack profiles are usually created based on the standard attack models and hence tend to show some kind of similarity between them. Many researchers have already mentioned that it is impossible for an attacker to have complete knowledge of the rating database.

Consequently, the attack profiles will differ statistically from those of genuine users. These differences and similarities can be expressed in terms of the attributes/features. Several attributes prevalent in the user profiles have been derived. The most common attack features generally used for detection are Rating Deviation from Mean Agreement (RDMA) [10], Weighted Deviation from Mean Agreement (WDMA) [18], Weighted Deviation from Agreement (WDA) [18], Length Variance (LENVAR) [19], Degree of Similarity (DEGSIM) [18], Filler Mean Target Difference (FMTD) [18], Filler Mean Difference (FMD), Mean-Variance (MEANVAR) [18], and Target Model Focus (TMF) [18]. These attributes tend to show different patterns in the case of genuine users and attackers. Thus, they have been successfully utilized in identifying whether a profile belongs to an authentic user or an attacker. Tab. 4 represents the notations used in the equations of the attribute that can be categorized into generic attributes and model-specific attributes, as shown in Tab. 5.

Table 4: Notations and their descriptions

Notation	Description	Notation	Description
N_u	The number of items rated by user u.	t_i	The number of ratings provided to item i by all the users.
K	The number of nearest neighbors.	$r_{u,i}$	The rating of i^{th} item given by user u.
l_u	Length of the profile of a user u.	$\overline{r_i}$	The average rating of item i over all users who have rated it.
P_u	The profile of a user u.	$\overline{l}$	The average length of a user's profile.
$P_{u,T}$	The set of target items of user u.	$\|P_u\|$	The number of ratings in the profile u.
$P_{u,F}$	The set of filler items of user u.	$Sim_{u,v}$	The similarity value between user u and v.
$\|P_{u,T}\|$	The number of target items of user u.	$\emptyset_{u,i}$	$\emptyset_{u,i} = 1$, if i $\in P_{u,T}$ and $\emptyset_{u,i} = 0$, otherwise.
$\|P_{u,F}\|$	The number of filler items of user u.		

Table 5: Rating-based attributes and their corresponding equations

Generic attributes		Model-specific attributes	
RDMA	$RDMA_u = \frac{\sum_{i=0}^{N_u} \left\| \frac{r_{u,i} - \overline{r_i}}{t_i} \right\|}{N_u}$	FMTD	$FMTD_u = \left(\frac{\sum_{i \in P_{u,T}} r_{u,i}}{\|P_{u,T}\|} \right) - \left(\frac{\sum_{k \in P_{u,F}} r_{u,k}}{\|P_{u,F}\|} \right)$
WDMA	$WDMA_u = \frac{\sum_{i=0}^{N_u} \left\| \frac{r_{u,i} - \overline{r_i}}{t_i^2} \right\|}{N_u}$	MEANVAR	$MeanVar_{P_{t,u}} = \frac{\sum_{i \in (P_u - P_{u,T})} (r_{i,u} - \overline{r_i})}{\|P_u\|}$

(Continued)

Table 5: Continued

Generic attributes		Model-specific attributes	
WDA	$WDA_u = \sum_{i=0}^{N_u} \left\| \frac{r_{u,i} - \overline{r_i}}{t_i} \right\|$	TMF	$F_i = \frac{\sum_{u \in U} \emptyset_{u,i}}{\sum_{u \in U} \|P_{u,T}\|}$
DEGSIM	$DegSim_u = \frac{\sum_{v=1}^{k} Sim_{u,v}}{k}$	LENVAR	$LenVar_u = \frac{\|l_u - \overline{l}\|}{\sum_{k \in U} (l_u - \overline{l})^2}$

2.3 Metrics Used to Check the Effectiveness of Shilling Attacks on Top-N Recommendations

Typically, three parameters are used to assess the effect of the attacks on the top-N recommendation of the target users, as discussed below.

Hit ratio: The hit ratio metric measures the average likelihood of a pushed item to be present in the top-N recommendation of a user [12]. Let, $R_u =$ top-N list and t $= \{t_1, t_2, .., t_k\}$ be the set of target items to be pushed, and U= set of users; then for every item t_i, hit ratio is calculated by Eq. (1).

$$H_r = \sum_{u \in U} \frac{H_{u,t_i}}{|U|} \tag{1}$$

where, $H_{u,t_i} = 1$, if $t_i \in R_u$ and $H_{u,t_i} = 0$, otherwise.

Biasing: In the context of the overall top-N list, a push attacker would want the I_T to be injected into the list, whereas a nuke attacker would like the I_T to be pulled down from the list. Let, t $= \{t_1, t_2, .., t_k\}$ be the set of target item to push/nuke and $T_a =$ overall top-N list after an attack is generated; then the biasing value is calculated using Eq. (2). A biasing value of 1 in a push attack means the entire target items have reached the top-N list, while in a nuke attack, it denotes that all the target items have been pulled down from the list.

$$B = \frac{\sum_{i=1}^{N} B_i}{N} \tag{2}$$

where, $B_i = 1$, if $t_i \in T_a$ and $B_i = 0$, otherwise.

Shuffling: Some attacks do not intend to push or nuke items but simply destroy the integrity of the recommendations. In the case of the overall top-N list, interchanging the positions of the items without injecting a target item deteriorates the accuracy of a recommendation system. The item ranked 1 is undoubtedly better and more preferred than the item ranked 3rd or 4th, even if they constitute the overall top-N list. Let, T be the overall top-N list before the attack and T_a be the top-N list generated after an attack is launched; then, shuffling can be calculated by Eq. (3). A shuffling value of 1 denotes that all positions and hence rankings of the overall top-N items have been interchanged.

$$S = \frac{\sum_{i=1}^{N} S_i}{N} \tag{3}$$

where, $S_i = 0$, if $T_i == T_{a_i}$ and $S_i = 1$, $T_i \neq T_{a_i}$.

3 Related Work

The objective of our designed model is to create attacks that bias the overall top-N list as well as the personalized top-N recommendation of every target user and escape from the feature-based detection. The attacks described are prone to detection due to their highly correlated nature [11–20]. Our model aims to reduce the DEGSIM. Instead of dropping the similarity value to -1, we just reduce it by rating the selected items with the least correlated ratings. The correlation between any two attackers based on these ratings is minimal. Hence, the combined effect of filler and selected items diminish the high correlation, i.e., rendering low DEGSIM values.

Paul et al. [21] mentioned that RDMA values for attackers would be high for attacks of small size, but once the attack size increases, several normal users show bigger RDMA values than the attackers. This is partly because an attack of such a large size is enough to radically increase the average rating for the target items (push) so that regular users who rated these items with the minimum rating would get an increased RDMA. Paul et al. also used the strategy of calculating DEGSIM first and segregating a set of users whose average similarity is less than half of the highest average similarity among users. The ratings of this set of users participate in calculating the average rating of each item. As the DEGSIM has been reduced in our model, the attack profiles also participate in calculating the average rating leading to a biased average value for the target items. The attack profiles do not deviate much from the average ratings rendering low RDMA values while computing. The variants of RDMA, WDMA, and WDA also give low values on account of this.

LENVAR has been considered a critical attribute in distinguishing real users from fake ones since very few real users would rate anything close to all the items available [19]. Since attackers hold no knowledge about the average length of a profile, they create lengthy profiles. It is highly anomalous that an authentic user would rate so many items manually. Attackers can still derive the number of items that are present in the system. We exploit this information to create profiles whose length is less than half the number of items in the system. This significantly reduces the LENVAR. We observe that reducing the filler items below a certain level reduces the attack effect, which means the attackers fail to become the nearest neighbors of the target users. Therefore, we constrain the number of filler items up to that level where the profile length is considerably less than the number of items, but it renders a good hit ratio at the same time. If the attack was only intended to bias the overall top-N items list, the profile length could be reduced to any length without a constraint as filler items do not play a major role in this context.

The attack feature called MEANVAR and its variant FMTD help to detect average attacks where the filler items are rated with a normal distribution centred around the mean rating of each filler item leading to very little difference between their ratings and the average ratings of the corresponding items.

For MEANVAR, every profile is searched for items with extreme ratings, which are labeled as target items for that profile. The remaining ones constitute the filler items. As target items in our model do not have extreme ratings, they will be labeled as filler items along with the selected items. The ratings of target and selected items will exhibit a greater deviation from their corresponding average ratings. Though the filler items will have less variance from their average ratings, the overall MEANVAR will increase due to the presence of the target and the selected items in the group.

For FMTD, every profile will be partitioned into $P_{u,T}$ and $P_{u,F}$ [18]. The difference between the target and filler items will be reduced since the target items in our model are not given extreme values, rendering low FMTD values for the attackers.

TMF works on the insight that an attack cannot be launched using a single attack profile. To push or nuke an item, several attack profiles are needed to rate the target item as a group. As already mentioned, the target items in our model will escape suspicion due to non-extreme ratings. In addition, the disguised items will play their part. Owing to their peak ratings, they get wrongly suspected and labeled as target items. The policy that two attackers can share disguised items, but no three attackers can share any, yields uncommon target items between attackers. This disrupts the effectiveness of this attribute.

Zhou et al. [22] suggested a method for detecting shilling attacks based on the assumption of a negative correlation among group users at various time periods. Lam and Riedl [16] investigated the answers to a few questions that could be useful in detecting attacks. The following are the questions: what algorithm is utilized to implement the recommendation system? Where is it utilized as an application? How can recommender system operators identify attacks? What properties do things with attackable properties have? Shriver et al. [23] developed a fuzzy-based model to test the recommender system's stability. For the evaluation of the recommendation system, Bansal and Baliyan [24] employed only segment attacks.

Our research differs significantly from the previously stated works. Instead of giving a method to detect existing shilling attacks, it provides a future opportunity to improve the efficacy or discover the limitations of the recommendation system by proposing a new shilling attack. Furthermore, unlike the aforementioned works that only use some of the attacks; this study examines feature-based shilling attacks as well as the unknown attack.

The closest to the presented work in this paper is of the Obfuscating attack where attackers inject their fallacious profiles into the recommendation system. These camouflaged profiles impersonate as genuine profiles, which confuse the attack detection schemes and are difficult to detect. Though the Obfuscating attack is very much effective to bias the recommendation process by disrupting the recommendation system, it is not so effective in a targeted attack.

4 Experimental Design of the Proposed Obscure Attack Model

Following the direction obtained from the concluding observations in Section 3, we designed a new attack model, where the popularly known attack features, as discussed in Section 2.2, lose their effectiveness in detecting it. We named the attack as the Obscure attack. In this section, we present the theoretical concepts and step-by-step procedure for generating the proposed Obscure attack, and afterward, we validate the effectivity of it.

4.1 Theoretical Consideration of the Obscure Attack

An Obscure attack consists of an additional set of items called the disguised items (I_D). The selection and ratings of the I_F and the I_S follow a different methodology from that of the standard attacks and the Obfuscated attack. Although our designed attack model hides the known attack features as the Obfuscated model does, it has a different design that aims to bias the overall top-N list of the target users. Tab. 6 identifies the notations used in the generation of Obscure attack, whereas Tab. 7 shows the structure of an Obscure attack model.

Table 6: List of notations

Notation	Description	Notation	Description
$T = \{t_{1,}\ t_{2,}\ldots,\ t_m\}$	A set of target items	T_{sim}	A set of items that are similar to the T items
$I_F = \{I_{F1,}\ I_{F2},\ldots, I_{FP}\}$	A set of filler items	U_T	A set of target users who have rated the T_{sim} items but not rated the T items
N	Number of attackers	T_{Dsim}	A set of items that are dissimilar to the T items computed using the Cosine Similarity
$D = H \cup L$	A set of disguised items	H	A set of items with high weighted average ratings
d_i	A set of disguised items for the i^{th}attacker	L	A set of items with low weighted average ratings

Table 7: An attack profile of an Obscure attack model

I_T	I_S	I_F	I_D	I_N
$I_1 \ldots I_t$	$I_1 \ldots I_s$	$I_1 \ldots I_f$	$I_1 \ldots I_d$	$I_1 \ldots I_n$

The algorithm for generating the Obscure attack is divided into the following steps:

1) **Target item and target user selection:** Our aim is to bias the overall top-N list. For this, some high-ranked items from the overall top-N list are selected as the target items in a nuke attack, and some low-ranked items are selected in a push attack. Unlike the standard attack profiles, which assign peak ratings to target items, we rate the target items randomly with 3 or 4 in a push attack, and similarly, the target items are rated randomly with 2 or 3 in a nuke attack. This strategy is followed as users assigning peak ratings are most likely to be suspected as attackers. In contrast, in the Obfuscated attack, the target shifting is done by reducing the rating of the target item by one step from the maximum rating in a push attack and raising the rating of the target item by one step in a nuke attack.
2) **Generation of filler items:** In order to become the nearest neighbors of the target users (U_T), the attackers need to mimic their behavior in rating the T_{sim} items. Hence, T_{sim} forms the I_F. The I_F for each attacker are selected and rated randomly from a normal distribution centered on the mean rating of each item. The T_{sim} items make the attackers appeared to be genuine users. The generation of target and filler item ratings is shown in Algorithm 1.

Algorithm 1: Generation of target and filler item ratings

```
Input: User-item rating dataset.
```

(Continued)

```
Output: List of target and filler item and ratings.
1. for i = 1 to n do
2.     for j = 1 to |T| do
3.         if $attack_{type}$ == PUSH
4.             $r_i$, $t_j$ = randomly selected from [3, 4] // Rating the target item for a push attack
5.         else
6.             $r_i$, $t_j$ = randomly selected from [2, 3] // Rating the target item for a nuke attack
7.     $I_F$ = randomly selected from $T_{sim}$ // Rating the filler items
8.     for k= 1 to $|I_F|$ do
9.         $r_i$, $I_{Fk}$ = randomly generated from N ( $\overline{r_{I_{Fk}}}$ , ${\sigma_{I_{Fk}}}^2$ )
```

3) **Generation of selected items:** The selected items play a significant role in our designed attack model. Since T_{Dsim} are dissimilar items, the attackers do not require to copy the behavior of U_T users in rating these items. Therefore, these items are utilized to create dissimilarity among the attackers. These items are rated in such a way so that the attackers have a minimum correlation between them based on the ratings of T_{Dsim} items. We use the Cholesky factorization to generate such least correlated ratings, as shown in Algorithm 2 [4].

Algorithm 2: Cholesky Factorization of the identity matrix

```
Input: Identity matrix of size (n)
Output: A lower triangular matrix L
1. Initialize $l_{11} = \sqrt{a_{11}}$
2. for k = 2, 3, …, n do
3.     for i = 1 to k-1 do
4.         $l_{ki} = \frac{a_{ki} - \sum_{j=1}^{i-1} l_{ij} l_{kj}}{l_{ii}}$
5.     $l_{kk} = \sqrt{\left(a_{kk} - \sum_{j=1}^{k-1} l^2_{kj}\right)}$
```

The purpose of the selected items was to reduce the similarity between the attackers. The attackers will have a zero correlation based on their ratings of selected items. Hence, the correlation matrix of attackers based on selected items will be an identity matrix I, denoting that every attacker has a zero correlation with other attackers and a correlation of 1 with itself. Every attacker has to rate all selected items; hence, the ratings of selected items will be generated as a matrix of size $|I_S|$ x n.

Let, n be the number of attackers and I_S be the set of items apart from the target and filler items whose average rating falls in the range of 2 and 3. It forms the set of selected items. The correlation matrix I of size n x n is a positive definite matrix and defined by Eq. (4).

$$I = LL^T \tag{4}$$

where, $I = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$ and $L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}$

A matrix, $Selected_{items}$ of size I_S x n is generated, where all ratings are randomly picked up from 2 and 3. Here, uncorrelated ratings of selected items, S = $Selected_{items} \times L$.

4) **Generation of disguised items:** To conceal the identity of the attackers, we create a set of disguised items to which the attackers assign extreme ratings. This set consists of some high-rated items and some least rated items apart from the target items. The high-rated items are rated with 5, while the least rated items are rated with 1. Every attacker rates not more than two or three disguised items. A strategy is followed where two attackers can share the disguised items, but no three attackers can share any such item. The procedure for generating the disguised items is given in Algorithm 3.

Algorithm 3: Generation of disguised items

```
Input: User-item rating dataset
Output: List of disguised item and rating
1.  for i = 1 to n do
2.      $d_i = \{\ \}$
3.      x = randomly pick up one value from {1, 2, 3}
4.      for j = 1 to x do
5.          d = randomly select one item from D
6.          $d_i = d_i \cup d$
7.          if i> 2 then
8.              for k = 1 to i do
9.                  for s = 1 to i do
10.                     if $(d_i \cap d_k \cap d_s) \neq \emptyset$ then
11.                         Goto step 10
```

4.2 Generating the Obscure Attack

To generate the proposed Obscure attack, we used the dataset given in Tab. 1. To attain the assumption of Section 1 (Scenario 2), we assume $I_T = \{I_1, I_2\}$, $I_F = \{I_5, I_6, I_7\}$, $I_S = \{I_8, I_9\}$, $I_D = \{I_3, I_4\}$, and $I_N = \{I_{10}\}$. The following steps are followed to generate an Obscure attack.

Step 1: All attackers give 2 or 3 ratings randomly to I_T. Tab. 8 shows the ratings of attackers on I_T.

Table 8: Ratings of target items

User	I_1	I_2
U_1	2	3
U_2	2	2
U_3	3	2
U_4	2	2

Step 2: Here, I_8 and I_9 are considered as the selected items. The following steps were follwed to generate the ratings of selected items.

a) Calculate the Correlation matrix (C), as shown in Tab. 9.
b) Generate a 2x4 matrix (no. of selected items = 2 and no. of attackers = 4) whose values are randomly in between the interval (1, 4).
c) Multiply the Cholesky factorization of C with the matrix generated in step (b).
d) Take the transpose of the resultant matrix, which is obtained in step (c). After that, we get the final ratings of selected items, as shown in Tab. 10.

Table 9: Correlation matrix of attackers

User	U_7	U_8	U_9	U_{10}
U_7	1	0	0	0
U_8	0	1	0	0
U_9	0	0	1	0
U_{10}	0	0	0	1

Table 10: Ratings of selected items

User	I_8	I_9
U_7	4	4
U_8	4	1
U_9	2	4
U_{10}	1	3

Step 3: Here, we consider I_5, I_6, and I_7 as the filler items. For item I_5, mean = 4, $\sigma = 0.57$, and the Gaussian distribution generates the ratings of attackers for I_5, as shown in Tab. 11. Similarly, we can compute the ratings of attackers for items I_6 and I_7.

Table 11: Ratings of filler items

User	I_5
U_7	5
U_8	4
U_9	5
U_{10}	4

Step 4: For the ratings of disguised items I_3 and I_4, we have to notice that no three attackers rate the same items. Attackers give a high rating to I_3 and low ratings to I_4 because based on the ratings, I_3 gets a greater number of high ratings than I_4. The resultant ratings of disguised items are shown in Tab. 12.

Table 12: Rating of disguised items

User	I_3	I_4
U_7	5	1
U_8	0	1
U_9	5	0
U_{10}	0	0

Step 5: However, I_{10} is considered in an I_N set; therefore, no attackers rate I_{10}.

After applying all steps (steps 1 to 5), the resultant ratings of all attackers for all items are shown in Tab. 13.

Table 13: Ratings of the attackers using Obscure attack

User	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
U_7	2	3	5	1	5	4	3	4	4	0
U_8	2	2	0	1	4	2	5	4	1	0
U_9	3	2	5	0	5	4	4	2	4	0
U_{10}	2	2	0	0	4	4	4	1	3	0

4.3 *Validation of the Proposed Obscure Attack*

To show the efficacy of the Obscure attack, Tab. 14 shows the top-3 similar users of the target items, and the recommended items, before and after the attack. From the table, we note that all target users have modified their top-3 similar neighbors after using Obscure attacks, while other shilling attacks fail. Other than that, in the Obscure attack, our predefined assumption is fulfilled.

Table 14: Top-3 similar neighbors of the target users and the recommended items before and after the Obscure attack

User	Top-3 similar neighbor		Recommended items	
	Authentic	Obscure attacked	Authentic	Obscure attacked
U_1	U_4, U_2, U_3	U_4, U_7, U_9		
U_2	U_1, U_4, U_5	U_7, U_1, U_9	I_7, I_{10}	I_9, I_7
U_3	U_4, U_1, U_2	U_4, U_1, U_{10}	I_3, I_8	I_3, I_8, I_9
U_4	U_1, U_3, U_2	U_1, U_3, U_9		

5 Experimental Analysis of the Effect of the Proposed Attack on Recommendation System

We collected the MovieLens dataset to provide a generalized solution that detects all types of attacks. The collected dataset is found to be attack-free due to the non-existence of any standard attack features. Therefore, it can be concluded that the MovieLens dataset contains only the ratings of the authentic users. This dataset consists of 943 users, 1682 movies, and 1,00,000 ratings [25–27]. These ratings are integer values between 1 and 5, where 1 denotes the lowest rating and 5 denotes the highest rating. Every user has given ratings to at least 20 movies.

The experimental results are divided into the following two phases:

a) Phase 1: In this phase, we show the biasing and shuffling effects of the standard attack model (such as random, average, bandwagon, and segment attack) and the new attacks (Obfuscated attack and our designed Obscure attack).

b) Phase 2: In this phase, we use the same attack size as like as phase 1 and compare the experimental results using Binary classifiers, trained with known attacks and tested with the known and unknown attacks.

The standard metrics such as precision and recall have been used to measure the performance of different attacks. These metrics are calculated using Eqs. (2) and (3).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

where, TP (true positive) is the number of attack profiles correctly classified as attackers, FP (false positive) is the number of authentic profiles wrongly classified as attackers, and FN (false negative) is the number of attack profiles wrongly classified as authentic users [13].

In addition to that, we computed the hit ratio values after the detection process has been performed.

5.1 Comparison of the Biasing, Shuffling, and Hit Ratio of Different Attacks on the Overall Top-N List

We generated different sizes of standard attacks of the push type. The biasing and shuffling effect of these different sizes of attacks create changes in the overall top-N list for recommendation. The attack sizes are gradually incremented to obtain the threshold t_s where the values of biasing, shuffling, and hit ratio become 1.

Fig. 2 compares the biasing, shuffling, and hit ratio of standard attacks on the overall top-N list. The biasing value and shuffling value of all standard attacks reach 1, at 25% and 20% attack sizes, respectively. We use a maximum 25% attack size for plotting the results of hit ratio and different filler sizes of standard attack because biasing and shuffling both values will reach 1. Hence, Fig. 2 represents the hit ratio of all attacks at different attack sizes across various filler sizes.

5.2 Comparative Analysis of Obscure Attack with Other Shilling Attacks

For training and testing, the dataset is divided into 60%–40%, respectively. In the training process, the binary classifiers are trained with the ratings given by authentic users and known attackers, which are labeled as authentic and attacker, respectively. The testing process contains the test set, which comprises ratings given by the authentic users and the known and unknown attackers.

The classifier detects the authentic users, attackers, and the effect of attacks on the top-N list of recommendations, on the test dataset using precision, recall, and hit ratio. Tab. 15 shows the precision, recall, and hit ratio of different classifiers on the detection of various attacks across four different filler sizes (such as 15%, 20%, 25%, and 40%). All classifiers provide a decent result in feature-based attack detection due to high precision, high recall values, and low hit ratio across four different filler sizes. But for the unknown attacks, all the classifiers attained low recall value and high hit ratio. Low recall value denotes the low detection rate of attackers from the test dataset, and high hit ratio represents the more effect on the top-N recommendation list. From Tab. 15, it can be observed that the Obscure attack provides more effect on the top-N list than the Obfuscated attack at 40% filler size for all the classifiers. The Obscure attack works better than the Obfuscated attack in escaping attack detection.

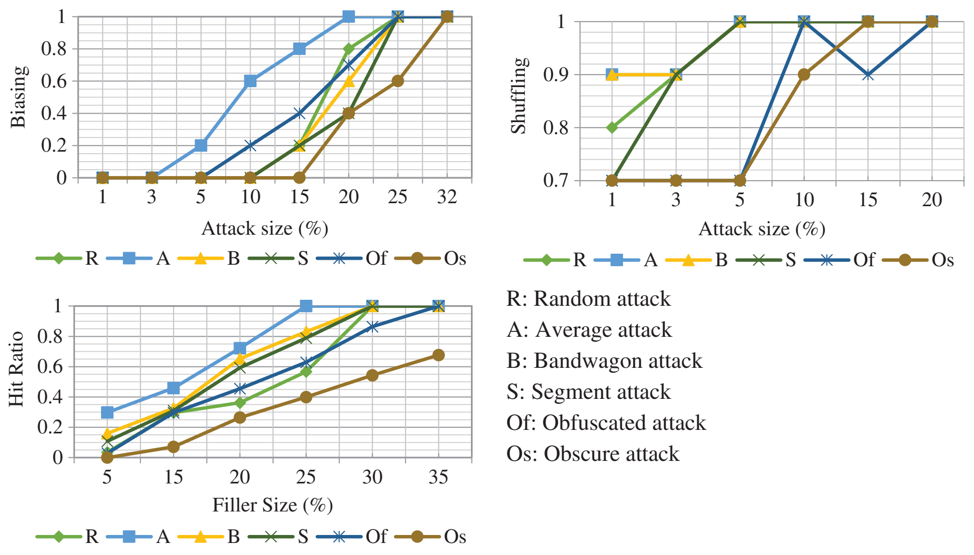

Figure 2: Biasing and Shuffling effects and the hit ratio of the standard attacks across various filler sizes on the overall top-N list

Table 15: Comparing detectability of Obscure attack with other attacks at different attack size across various filler sizes based on precision (P), recall (R), and hit ratio (H)

Classifier	Filler size											
	15%			20%			25%			40%		
	P	R	H	P	R	H	P	R	H	P	R	H
Random attack (25% attack size across various filler sizes)												
SVM	0.989	0.990	0	0.973	0.991	0	0.979	0.982	0	0.995	0.98	0
J48	0.99	0.992	0	0.992	0.991	0	0.991	0.991	0	0.993	0.991	0
Random Forest	0.978	0.981	0	0.985	0.996	0	0.997	0.976	0	0.982	0.975	0
Naïve Bayes	0.98	0.989	0	0.997	0.996	0	0.977	0.989	0	0.991	0.992	0
Average attack (25% attack size across various filler sizes)												
SVM	0.952	0.968	0	0.972	0.989	0	0.981	0.979	0	0.993	0.978	0
J48	0.992	0.991	0	0.992	0.991	0	0.992	0.991	0	0.992	0.991	0
Random Forest	0.98	0.979	0	0.983	1	0	0.995	0.971	0	0.98	0.972	0
Naïve Bayes	1	1	0	0.996	0.994	0	0.974	0.987	0	0.993	0.990	0

(Continued)

Table 15: Continued

Classifier	Filler size											
	15%			20%			25%			40%		
	P	R	H	P	R	H	P	R	H	P	R	H
Bandwagon attack (25% attack size across various filler sizes)												
SVM	0.975	0.980	0	0.972	0.984	0	0.98	0.982	0	0.988	0.980	0
J48	0.990	0.991	0	0.992	0.993	0	0.989	0.990	0	0.992	0.993	0
Random Forest	0.978	0.98	0	0.983	0.986	0	0.991	0.977	0	0.983	0.986	0
Naïve Bayes	0.997	0.989	0	0.996	0.993	0	0.98	0.992	0	0.995	0.990	0
Segment attack (25% attack size across various filler sizes)												
SVM	0.982	0.97	0	0.972	0.989	0	0.982	0.978	0	0.995	0.980	0
J48	0.991	0.990	0	0.992	0.991	0	0.993	0.991	0	0.990	0.993	0
Random Forest	0.983	0.978	0	0.989	0.991	0	0.996	0.973	0	0.979	0.975	0
Naïve Bayes	0.998	0.997	0	0.995	0.996	0	0.987	0.989	0	0.993	0.992	0
Obfuscated attack (28% attack size across various filler sizes)												
SVM	0.685	0.440	0.205	0.695	0.554	0.219	0.742	0.578	0.321	0.797	0.612	0.482
J48	0.652	0.471	0.150	0.679	0.523	0.223	0.712	0.567	0.352	0.732	0.689	0.451
Random Forest	0.669	0.461	0.198	0.691	0.523	0.245	0.706	0.527	0.381	0.729	0.678	0.457
Naïve Bayes	0.699	0.434	0.212	0.721	0.511	0.261	0.745	0.586	0.319	0.789	0.611	0.487
Obscure attack (32% attack size across various filler sizes)												
SVM	0.824	0.239	0.012	0.830	0.253	0.215	0.833	0.269	0.311	0.837	0.297	0.571
J48	0.841	0	0.095	0.850	0	0.3	0.852	0.131	0.451	0.853	0.207	0.602
Random Forest	0.835	0	0.095	0.841	0	0.3	0.839	0.143	0.424	0.845	0.213	0.593
Naïve Bayes	0.821	0.267	0.007	0.833	0.275	0.198	0.837	0.291	0.295	0.840	0.313	0.520

For further evaluation, we computed the accuracy of the attack detection when the models are trained and tested with the ratings of users under various scenarios. The accuracy (%) of attack detection is defined by Eq. (7).

$$\text{Accuracy} = \frac{\begin{array}{c}\textit{No. of authentic users and}\\ \textit{correctly classified attackers}\end{array}}{\textit{Total no. of users in the test set}} \times 100 \tag{7}$$

Fig. 3 portrays the accuracy of four different classifiers on the feature-based detection of different known and unknown attacks. It can be clearly observed that all classifiers obtain high detection accuracy on feature-based detection schemes at different filler sizes, whereas the detection accuracy is significantly decreased in case of unknown attacks. It can be observed that the detection accuracy of the Obscure attack is lowest compared to all other attacks.

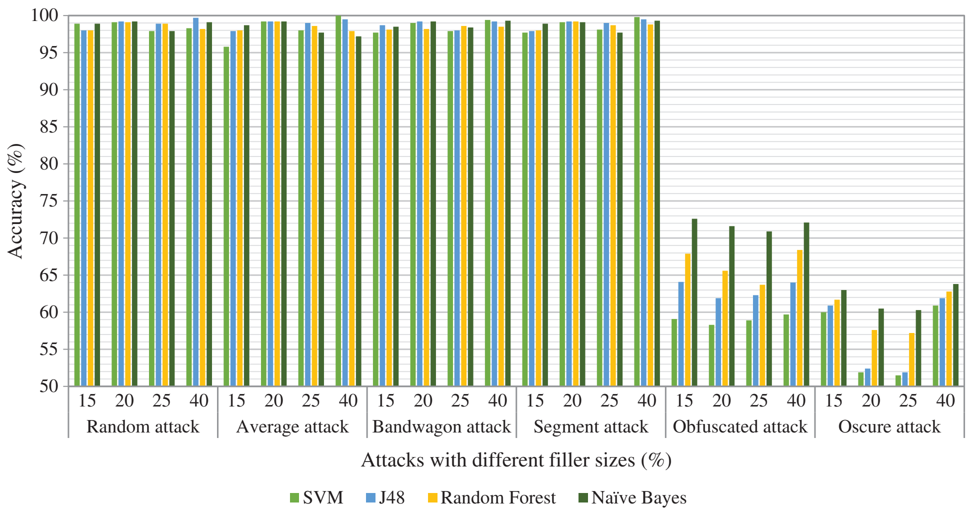

Figure 3: Detection accuracy of different attacks (known and unknown)

6 Conclusion and Future Work

Due to their effectiveness and popularity, recommendation systems have been the target of the attackers. They use shilling attacks to disrupt the list of recommendable items. The malicious user profiles created by the attackers closely match with the real users. To keep the recommendation systems reliable, the authenticity and security of these systems are very crucial.

In this paper, we generated a new attack model with unknown features and proved that though the existing shilling attack detection methods can detect the standard shilling attacks, they fail to detect the new attacks with unknown features. Our proposed attack is more effective compared to other existing attacks.

This paper measures the rating pattern of the attackers and the authentic users based on standard attack features. We considered only three attackers with the same ratings for a particular item. Due to this, our proposed Obscure attack may have little more computational complexity compared to other standard shilling attacks. However, this should not be a serious issue because the attacker's profiles are usually generated offline.

In future, two possible solutions to improve the detection of unknown attacks can be thought of. The first is to train the classifier with the ratings of the trustworthy users, which can be obtained by applying different attack features on the rating dataset. A user profile will be considered malicious if it does not belong to the class of trustworthy users. The second is to use rated item correlation as a feature in attack detection. In this case, a user may be considered as the attacker if its rated item correlation is different from the real users.

Funding Statement: Funding is provided by Taif University Researchers Supporting Project number (TURSP-2020/10), Taif University, Taif, Saudi Arabia.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. Chen, P. P. Chan, F. Zhang and Q. Li, “Shilling attack based on item popularity and rated item correlation against collaborative filtering,” *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 7, pp. 1833–1845, 2019.

[2] P. Cremonesi, Y. Koren and R. Turrin, “Performance of recommender algorithms on top-n recommendation tasks,” in *Proc. Fourth ACM Conf. on Recommender Systems (RecSys ‘10)*, Barcelona, Spain, 2010.

[3] G. Karypis, “Evaluation of item-based top-n recommendation algorithms,” in *Proc. 10th Int. Conf. on Information and Knowledge Management (CIKM ‘01)*, Atlanta, Georgia, USA, 2001.

[4] E. Christakopoulou and G. Karypis, “Local item-item models for top-n recommendation,” in *Proc. 10th ACM Conf. on Recommender Systems (RecSys ‘16)*, Boston, USA, 2016.

[5] P. K. Singh, P. K. D. Pramanik and P. Choudhury, “A comparative study of different similarity metrics in highly sparse rating dataset,” in *Proc. 2nd Int. Conf. on Data Management, Analytics and Innovation (ICDMAI 2018). Advances in Intelligent Systems and Computing*, vol. 839, Springer, Singapore, pp. 45–60, 2019.

[6] M. Deshpande and G. Karypis, “Item-based top-n recommendation algorithms,” *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 143–177, 2004.

[7] R. A. Zayed, L. F. Ibrahim, H. A. Hefny and H. A. Salman, “Shilling attacks detection in collaborative recommender system: Challenges and promise,” in *Web, Artificial Intelligence and Network Applications (WAINA 2020). Advances in Intelligent Systems and Computing*, vol. 1150, L. Barolli, F. Amato, F. Moscato, T. Enokido and M. Takizawa, (Eds.), Springer, Cham, pp. 429–439, 2020.

[8] C. Williams, B. Mobasher, R. Burke, J. Sandvig and R. Bhaumik, “Detection of obfuscated attacks in collaborative recommender systems,” in *Proc. ECAI’06 Workshop on Recommender Systems (17th European Conf. on Artificial Intelligence (ECAI’06))*, Riva del Garda, Italy, 2006.

[9] P. K. Singh, P. K. D. Pramanik and P. Choudhury, “Collaborative filtering in recommender systems: Technicalities, challenges, applications, and research trends,” in *New Age Analytics: Transforming the Internet Through Machine Learning, IoT, and Trust Modeling*, G. Shrivastava, S. L. Peng, H. Bansal, K. Sharma and M. Sharma, (Eds.), Apple Academic Press, Burlington, Canada, pp. 183–215, 2020.

[10] M. P. O’Mahony, N. J. Hurley and G. C. Silvestre, “Recommender systems: Attack types and strategies,” in *Proc. 12th National Conf. on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conf.*, Pittsburgh, USA, 2005.

[11] Z. Zhang and S. R. Kulkarni, “Graph-based detection of shilling attacks in recommender systems,” in *Proc. IEEE Int. Workshop on Machine Learning for Signal Processing*, Southampton, UK, 2013.

[12] B. Mobasher, R. Burke, R. Bhaumik and J. J. Sandvig, “Attacks and remedies in collaborative recommendation,” *IEEE Intelligent Systems*, vol. 22, no. 3, pp. 56–63, 2007.

[13] B. Mobasher, R. Burke, R. Bhaumik and C. Williams, “Effective attack models for shilling item-based collaborative filtering systems,” in *Proc. WebKDD Workshop*, Chicago, USA, 2005.

[14] B. Mobasher, R. Burke, R. Bhaumik and C. Williams, “Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness,” *ACM Transactions on Internet Technology*, vol. 7, no. 4, pp. 23:1–23:38, 2007.

[15] P. Kaur and S. Goel, “Shilling attack models in recommender system,” in *Proc. Int. Conf. on Inventive Computation Technologies*, Coimbatore, India, 2016.

[16] S. K. Lam and J. Riedl, “Shilling recommender systems for fun and profit,” in *Proc. 13th Int. Conf. on World Wide Web (WWW ‘04)*, New York, USA, 2004.

[17] R. Burke, B. Mobasher, R. Bhaumik and C. Williams, “Segment-based injection attacks against collaborative filtering recommender systems,” in *Proc. 5th IEEE Int. Conf. on Data Mining*, Houston, USA, 2005.

[18] R. Burke, B. Mobasher, C. Williams and R. Bhaumik, “Classification features for attack detection in collaborative recommender systems,” in *Proc. 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Philadelphia, USA, 2006.

[19] W. Bhebe and O. P. Kogeda, "Shilling attack detection in collaborative recommender systems using a meta learning strategy," in *Proc. Int. Conf. on Emerging Trends in Networks and Computer Communications*, Windhoek, Namibia, 2015.

[20] Z. Zhang and S. R. Kulkarni, "Detection of shilling attacks in recommender systems via spectral clustering," in *Proc. 17th Int. Conf. on Information Fusion*, Salamanca, Spain, 2014.

[21] P. A. Chirita, W. Nejdl and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Proc. 7th Annual ACM Int. Workshop on Web Information and Data Management*, Bremen, Germany, 2005.

[22] W. Zhou, J. Wen, Q. Qu, J. Zeng and T. Cheng, "Shilling attack detection for recommender systems based on credibility of group users and rating time series," *PLOS One*, vol. 13, no. 5, pp. e0196533, 2018.

[23] D. Shriver, S. Elbaum, M. B. Dwyer and D. S. Rosenblum, "Evaluating recommender system stability with influence-guided fuzzing," *Proc. AAAI Conf. on Artificial Intelligence*, vol. 33, no. 1, pp. 4934–4942, 2019.

[24] S. Bansal and N. Baliyan, "Evaluation of collaborative filtering based recommender systems against segment-based shilling attacks," in *Proc. Int. Conf. on Computing, Power and Communication Technologies (GUCON)*, New Delhi, India, 2019.

[25] P. K. Singh, S. Setta, P. K. D. Pramanik and P. Choudhury, "Improving the accuracy of collaborative filtering-based recommendations by considering the temporal variance of top-n neighbors," in *Proc. Int. Conf. on Innovative Computing and Communication (ICICC-2019). Advances in Intelligent Systems and Computing*, vol. 1087, Springer, Singapore, pp. 1–10, 2020.

[26] P. K. Singh, P. K. D. Pramanik, N. C. Debnath and P. Choudhury, "A novel neighborhood calculation method by assessing users' varying preferences in collaborative filtering," in *Proc. 34th Int. Conf. on Computers and Their Applications*, Honolulu, Hawaii, 2019.

[27] P. K. Singh, M. Sinha, S. Das and P. Choudhury, "Enhancing recommendation accuracy of item-based collaborative filtering using bhattacharyya coefficient and most similar item," *Applied Intelligence*, vol. 50, no. 12, pp. 4708–4731, 2020.