Effective Attack Models for Shilling Item-Based Collaborative Filtering Systems

Bamshad Mobasher, Robin Burke, Runa Bhaumik, Chad Williams
Center for Web Intelligence
School of Computer Science, Telecommunication and Information Systems
DePaul University, Chicago, Illinois
{mobasher,rburke,rbhaumik,cwilli}@cs.depaul.edu

ABSTRACT

Significant vulnerabilities have recently been identified in collaborative filtering recommender systems. These vulnerabilities mostly emanate from the open nature of such systems and their reliance on user-specified judgments for building profiles. Attackers who cannot be readily distinguished from ordinary users may introduce biased data in an attempt to force the system to "adapt" in a manner advantageous to them. A handful of simple attack models have, so far, been identified, and there appear to be significant differences in the susceptibility of different recommendation techniques to these attacks. In particular, item-based collaborative filtering has been found to offer some security advantages over user-based collaborative filtering. Our research in secure personalization is examining a range of more complex attack models and recommendation techniques, paying particular attention to the costs and benefits of mounting an attack. In this paper, we take a closer look at item-based collaborative filtering. In particular, we propose a new attack model that focuses on a subset of users with similar tastes and show that such an attack can be highly successful against an item-based algorithm.

Key Words: Shilling, Collaborative Filtering, Recommender Systems, Attack Models

1. INTRODUCTION

Recent research has begun to examine the vulnerabilities and robustness of different recommendation techniques, such as collaborative filtering, in the face of what has been termed "shilling" attacks [2, 1, 5, 6]. Attackers who cannot be readily distinguished from ordinary users may introduce biased data in an attempt to force the system to "adapt" in a manner advantageous to them. Recommendation systems, as well as many other user-adaptive systems are vulnerable to such attacks, precisely because they rely on users' interactions with the system and past user profiles to generate recommendations or dynamic content. The wide-spread use of such systems in domains such as electronic commerce and information access provides a strong motivation for unscrupulous agents to use such attacks in the hope of gaining economic advantage.

It is easy to see why collaborative filtering is vulnerable to shilling attacks. A user-based collaborative filtering algorithm collects user profiles, which are assumed to represent the preferences of many different individuals and makes recommendations by finding peers with like profiles. If the profile database contains biased data (many profiles all of which rate a certain item highly, for example), these biased profiles may be considered peers for genuine users and result in biased recommendations. This is precisely the effect found in [5] and [6]. We have replicated these results and begun to extend them to consider alternative attack models.

Our work considers in particular the cost of mounting an attack. This cost has two primary components: knowledge cost and execution cost. Knowledge cost is the cost or effort required to gather information about the system to be attacked or its users. We assume that the more detailed the knowledge that is required by an attack (details of the rating distribution across profiles, for example) the more costly the attack will be to mount. The execution cost is the effort required in terms of interactions with the system to add the necessary profiles and ratings to execute the attack. While this latter cost may seem irrelevant when automated software agents can generate the needed profiles, we believe that it remains a relevant consideration. To defend against shilling attacks, sites may implement policies limiting the speed with which profiles can be built. Thus, an attack that requires a small number of short attack profiles would be much more practical to mount and more difficult to detect and defend against than one that requires that many profiles be constructed, each with many ratings.

Lam et al. [5] show that item-based collaborative filtering appears to offer an advantage over the user-based approach. In item-based collaborative filtering, the system looks for items with similar profiles and makes predictions based on a user's own rating of these peer items (see example below). By adding biased user profiles, an attacker can only alter a portion of the profile for any given item. In [2], we suggested that an attack could be designed specifically to target an item-based recommendation algorithm if it is designed to

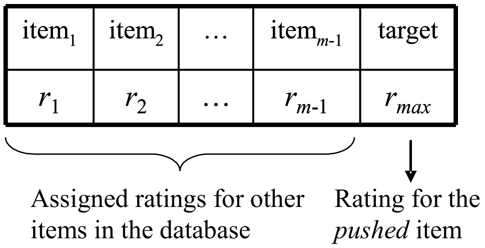


Figure 1: The general form of a push attack profile.

change the distances between item profiles in specific ways. This attack, called here the favorite item attack, is designed to target individual users by co-rating their favorite items with a target item. However, such an attack presents too significant a knowledge requirement to be of practical use by an attacker. In order to know which items are the best peers for the target item, the attacker must know what each user's ratings are for each item.

This paper proposes a generalization of the favorite item attack called the segmented attack. A segmented attack is one that pushes an item to a targeted group of users with known or easily predicted preferences. Profiles are inserted that maximize the similarity between the pushed item and items preferred by the group. As we show below, the segmented attack is both effective and practical against standard item-based collaborative filtering algorithms.

The paper is organized as follows. In Section 2 we provide a detailed description of various attack models against collaborative filtering systems, including those proposed in earlier work, as well as some that we have examined in out research. Section 3 includes some background information and the specific details of the user-based and item-based recommendation algorithms used in our experiments. This section also contains a description of the evaluation metrics we have used to determine the effectiveness of various attack models. In Section 4 we present our experimental results. We first show the impact of some of the previously studied attack models on the item-based algorithm. We then provide a detailed analysis of the proposed segmented attack model and experimentally show that it can be effective against item-based collaborative filtering.

2. ATTACK MODELS

An attack against a collaborative filtering recommender system consists of a set of attack profiles, biased profile data associated with fictitious user identities, and a target item, the item that the attacker wishes the system to recommend more highly (a *push* attack), or wishes to prevent the system from recommending (a *nuke* attack). We concentrate on push attacks in this paper. An attack model is an approach to constructing the attack profile, based on knowledge about the recommender system, its rating database, its products, and/or its users. The general form of a push attack profile is depicted in Figure 1. An attack profile consists of an m-dimensional vector of ratings, were m is the total number of items in the system. The rating given to the pushed item, *target*, is r_{max} and is the maximum allowable rating value. On the other hand, the ratings r_1 through r_{m-1} are assigned to the corresponding items according to the specific attack model. Indeed, the specific strategy used to assign ratings to items 1 through $m - 1$ is what determines the type of attack model used.

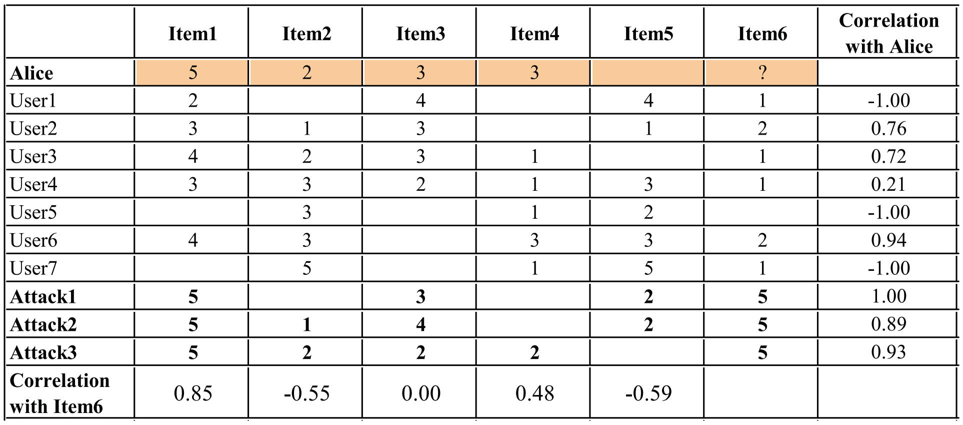
In the remainder of this section, we provide an illustrative example that will help illustrate the vulnerability of collaborative filtering algorithms, and will serve as a motivation for the attack models, which we will then describe more formally.

2.1 An Example

Consider, as an example, a recommender system that identifies books that users might like to read using a user-based collaborative algorithm [3]. A user profile in this hypothetical system might consist of that user's ratings (in the scale of 1-5 with 1 being the lowest) on various books. Alice, having built up a profile from previous visits, returns to the system for new recommendations. Figure 2 shows Alice's profile along with that of seven genuine users. An attacker, Eve, has inserted attack profiles (Attack1-3) into the system, all of which give high ratings to her book labeled Item6. Eve's attack profiles may closely match the profiles of one or more of the existing users (if Eve is able to obtain or predict such information), or they may be based on average or expected ratings of items across all users.

If the system is using a standard user-based collaborative filtering approach, then the predicted ratings for Alice on Item6 will be obtained by finding the closest neighbors to Alice. Without the attack profiles, the most similar user to Alice, using correlation-based similarity, would be User6. The prediction associated with Item6 would be 2, essentially stating that Item6 is likely to be disliked by Alice. After the attack, however, the Attack1 profile is the most similar one to Alice, and would yield a predicted rating of 5 for Item6, the opposite of what would have been predicted without the attack. So, in this example, the attack is successful, and Alice will get Item6 as a recommendation, regardless of whether this is really the best suggestion for her. She may find the suggestion inappropriate, or worse, she may take the system's advice, buy the book, and then be disappointed by the delivered product.

On the other hand, if a system is using an item-based collaborative filtering approach, then the predicted rating for Item6 will be determined by comparing the rating vector for Item6 with those of the other items. This algorithm does not lend itself to an attack as obvious as the previous one, since Eve does not have control over ratings given by other users to any given item. However, if Eve can obtain some knowledge about the rating distributions for some items, this can make a successful attack more likely. In the example of Figure 2, for instance, Eve knows that Item1 is a popular item among a significant group of users to which Alice also belongs. By designing the attack profiles so that high ratings are associated with both Item1 and Item6, Eve can attempt to increase the similarity of these two items, resulting in a higher likelihood that Alice (and the rest of the targeted group) will receive Item6 as a recommendation. Indeed, as the example portrays, such an attack is highly successful regardless of whether the system is using an item-based or a user-based algorithm. This latter observation illustrates



	Item1	Item2	Item3	Item4	Item5	Item6	Correlation with Alice
Alice	5	2	3	3		?	
User1	2		4		4	1	-1.00
User2	3	1	3		1	2	0.76
User3	4	2	3	1		1	0.72
User4	3	3	2	1	3	1	0.21
User5		3		1	2		-1.00
User6	4	3		3	3	2	0.94
User7		5		1	5	1	-1.00
Attack1	**5**		**3**		**2**	**5**	1.00
Attack2	**5**	**1**	**4**		**2**	**5**	0.89
Attack3	**5**	**2**	**2**	**2**		**5**	0.93
Correlation with Item6	0.85	-0.55	0.00	0.48	-0.59		

Figure 2: An example of a push attack favoring the target item Item6.

the motivation behind the attack model we introduce and analyze in this paper, namely the segmented attack.

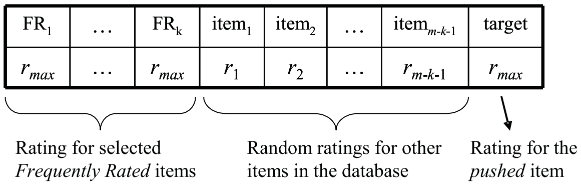
2.2 Attack Models

Prior work on recommender system stability has examined primarily three types of attack models:

- **Sampling attack:** A sampling attack is one in which attack profiles are constructed from entire user profiles sampled from the actual profile database, augmented by a positive rating for the pushed item. This attack is used by O'Mahony et al. [6] to provide a proof of the instability of collaborative filtering algorithms, but is the least practical from a knowledge cost standpoint.

- **Random attack:** Lam et al. [5] show an attack model in which profiles consist of random values (except of course for a positive rating given to the pushed item). Specifically, r_1 through r_{m-1} are assigned to the corresponding items by generating random values within the rating scale with a distribution centered around the mean for all user ratings across all items (see Figure 1). The knowledge required to mount such an attack is quite minimal, especially since the overall rating mean in many systems can be determined by an outsider empirically (or, indeed, may be available directly from the system). The execution cost involved, however, is still substantial, since it involves assigning ratings to every item in each attack profile. Furthermore, as [5] shows and our results confirm [1], the attack is not particularly effective.

- **Average attack:** A more powerful attack described in [5] uses the individual mean for each item rather than the global mean (except again the pushed item.) In the average attack, each assigned rating, r_i, in an attack profile corresponds (either exactly or approximately) to the mean rating for $item_i$, across the users in the database who have rated that item (see Figure 1). In addition to the effort involved in producing the ratings, the average attack also has considerable knowledge cost of order m. Our experiments, however, have shown that, in the case of user-based algorithm, the average attack can be just as successful by assigning the average ratings to a small subset of items in the database, thus substantially reducing the knowledge requirement [1]. This attack model, however, is not, in general, effective against an item-based collaborative algorithm, as show in Section 4 below.



FR_1	…	FR_k	$item_1$	$item_2$	…	$item_{m\text{-}k\text{-}1}$	target
r_{max}	…	r_{max}	r_1	r_2	…	$r_{m\text{-}k\text{-}1}$	r_{max}

Rating for selected *Frequently Rated* items

Random ratings for other items in the database

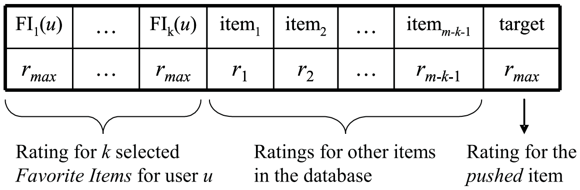
Rating for the *pushed* item

Figure 3: A *Bandwagon* attack profile.

In addition to these attack models, we have introduced several others that are described below. Some of these attack models were introduced in [2] and were analyzed in the context of user-based collaborative filtering in [1]. In this paper, we discuss these attacks in the context of item-based collaborative filtering.

- **Bandwagon attack:** This attack takes advantage of the Zipf's law distribution of popularity in consumer markets: a small number of items, best-seller books for example, will receive the lion's share of attention and also ratings. The attacker using this model will build attack profiles containing those items that have high visibility. Such profiles will have a good probability of being similar to a large number of users, since the high visibility items are those that many users have rated. For example, by associating her book with current best-sellers, for example, *The DaVinci Code*, Eve can ensure that her bogus profiles have a good probability of matching any given user, since so many users will have these items on their profiles. This attack can be considered to have low knowledge cost. It does not require any system-specific data, because it is usually not difficult to independently determine what the "blockbuster" products are in any product space.

  Figure 3 depicts a typical attack profile for the bandwagon attack. Items FR_1 through FR_k are selected



$FI_1(u)$	…	$FI_k(u)$	$item_1$	$item_2$	…	$item_{m-k-1}$	target
r_{max}	…	r_{max}	r_1	r_2	…	$r_{m\text{-}k\text{-}1}$	r_{max}

Rating for *k* selected *Favorite Items* for user *u*
Ratings for other items in the database
Rating for the *pushed* item

Figure 4: A *Favorite Item* attack profile.

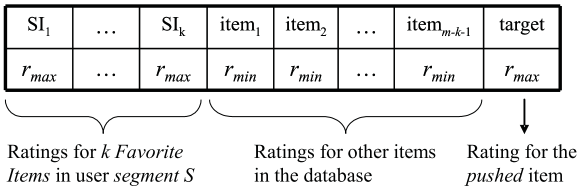
because they have been rated by a large number of users in the database. These items are assigned the maximum rating value together with the target item. The ratings r_1 through r_{m-k-1} for the other items are determined randomly in a similar manner as in the random attack. The bandwagon attack therefore can be viewed as an extension of the random attack. We showed in [4] that the bandwagon attack can still be successful even when only a small subset of the "random items", $item_1$ through $item_{m-k-1}$ are assigned ratings. However, as in the case of the average attack, it falls short when used against an item-based algorithm, as shown in Section 4 below.

- **Favorite item attack:** (called the "consistency attack" in [2]) Rather than knowledge about items, the favorite item attack looks at knowledge of user's preferences. Such an attack is mounted not against the system as a whole, but by targeting a given user. We assume that the attacker knows which items a given user, u, really likes, and builds profiles containing only those items. Like the sampling attack, this attack is not particularly practical from a knowledge cost standpoint, but provides an upper bound on the effectiveness of other attacks focused on user characteristics.

  Figure 4 depicts a typical attack profile for the favorite item attack. $FI_i(u)$ represent the favorite items by user u selected in the attack profile. These favorite items are the ones whose ratings are greater than the user's average rating. These items are assigned maximum rating value together with the target item. The other items in the database, $item_1$ through $item_{m-k-1}$ are assigned ratings at random or based on other criteria. In our experiments, best results were obtained when the non-favored items are assigned the lowest possible rating. Given its direct tailoring to each user, it is not surprising that the favorite item attack is effective against both user-based and item-based algorithms as our experiments have suggested [1].

- **Segmented attack:** The segmented attack is a generalization of the favorite item attack. It may be impossible to know what items are preferred by a given user, but it is possible to discover what items are well liked by a targeted segment of users and use this fact to attack that segment specifically. In fact, such an approach is probably one with great pragmatic appeal to an attacker. For example, if Eve were an author of a fantasy book for children, she would probably much prefer to have her book pushed to users who are fans of the Harry Potter series than to readers of gardening books.

  This attack also requires very limited knowledge about the system and the users. An attacker needs to know only a group of items well liked by the target segment and needs to build profiles containing only those items. Figure 5 depicts a typical attack profile for the segmented attack. Items SI_1 through SI_k are the specific items, in our case they are the movies common to a segment of users. These items are assigned the maximum rating value together with the target item. The ratings r_1 through r_{m-k-1} are assigned ratings at random or based on other criteria. As with the favorite item attack, the best results were obtained when these items are assigned to 1, the lowest possible rating.



SI_1	…	SI_k	$item_1$	$item_2$	…	$item_{m-k-1}$	target
r_{max}	…	r_{max}	r_{min}	r_{min}	…	r_{min}	r_{max}

Ratings for *k Favorite Items* in user *segment S*
Ratings for other items in the database
Rating for the *pushed* item

Figure 5: A *Segmented* attack profile.

3. RECOMMENDATION ALGORITHMS AND EVALUATION METRICS

We have concentrated in this work on the most commonly-used algorithms for collaborative filtering. Each algorithm assumes that there is a user / item pair for whom a prediction is sought, the target user and the target item. The task for the algorithm is to predict the target user's rating for the target item.

3.1 User-Based Collaborative Filtering

The standard collaborative filtering algorithm is based on user-to-user similarity [3]. This kNN algorithm operates by selecting the k most similar users to the target user, and formulates a prediction by combining the preferences of these users. kNN is widely used and reasonably accurate. The similarity between the target user, u, and a neighbor, v, can be calculated by the Pearson's correlation coefficient defined below:

$$sim_{u,v} = \frac{\sum\limits_{i \in I} (r_{u,i} - \bar{r}_u) * (r_{v,i} - \bar{r}_v)}{\sqrt{\sum\limits_{i \in I} (r_{u,i} - \bar{r}_u)^2} * \sqrt{\sum\limits_{i \in I} (r_{v,i} - \bar{r}_v)^2}}$$

where I is the set of all items that can be rated, $r_{u,i}$ and $r_{v,i}$ are the ratings of some item i for the target user u and a neighbor v, respectively, and $\bar{r}_u$ and $\bar{r}_v$ are the average of the ratings of u and v over I, respectively. Once similarities are calculated, the most similar users are selected. In our implementation, we have used a value of 20 for the neighborhood size k. We also filter out all neighbors with a similarity of less than 0.1 to prevent predictions being based on very distant or negative correlations. Once the most similar users are identified, we use the following formula to compute the prediction for an item i for target user u.

$$p_{u,i} = \bar{r}_a + \frac{\sum\limits_{v \in V} sim_{u,v}(r_{v,i} - \bar{r}_v)}{\sum\limits_{v \in V} |sim_{u,v}|}$$

where V is the set of k similar users and $r_{v,i}$ is the rating of those users who have rated item i, $\bar{r}_v$ is the average rating for the target user over all rated items, and $sim_{u,v}$ is the mean-adjusted Pearson correlation described above. The formula in essence computes the degree of preference of all the neighbors weighted by their similarity and then adds this to the target user's average rating: the idea being that different users may have different "baselines" around which their ratings are distributed.

3.2 Item-Based Collaborative Filtering

Item-based collaborative filtering works by comparing items based on their pattern of ratings across users. Again, a nearest-neighbor approach can be used. The kNN algorithm attempts to find k similar items that are co-rated by different users similarly.

For our purpose we have adopted the adjusted cosine similarity measure introduced by [7]. The adjusted cosine similarity formula is given by:

$$sim_{i,j} = \frac{\sum\limits_{u \in U} (r_{u,i} - \bar{r}_u) * (r_{u,j} - \bar{r}_u)}{\sqrt{\sum\limits_{u \in U} (r_{u,i} - \bar{r}_u)^2} * \sqrt{\sum\limits_{u \in U}^{n} (r_{u,j} - \bar{r}_u)^2}}$$

where $r_{u,i}$ represents the rating of user u on item i, and $\bar{r}_u$ is the average of the user u's ratings as before. After computing the similarity between items we select a set of k most similar items to the target item and generate a predicted value by using the following formula:

$$p_{u,i} = \frac{\sum\limits_{j \in J} r_{u,j} * sim_{i,j}}{\sum\limits_{j \in J} sim_{i,j}}$$

where J is the set of k similar items, $r_{u,j}$ is the prediction for the user on item j, and $sim_{i,j}$ is the similarity between items i and j as defined above. We consider a neighborhood of size 20 and ignore items with negative similarity. The idea here is to use the user's own ratings for the similar items to extrapolate the prediction for the target item.

3.3 Evaluation Metrics

There has been considerable research in the area of recommender systems evaluation [4]. Some of these concepts can also be applied to the evaluation of the security of recommender systems, but in evaluating security, we are interested not in raw performance, but rather in the change in performance induced by an attack. In [6] two evaluation measures were introduced: *robustness* and *stability*. Robustness measures the performance of the system before and after an attack to determine how the attack affects the system as a whole. Stability looks at the shift in system's ratings for the attacked item induced by the attack profiles.

Our goal is to measure the effectiveness of an attack - the "win" for the attacker. The desired outcome for the attacker in a "push" attack is of course that the pushed item be more likely to be recommended after the attack than before. In the experiments reported below, we follow the lead of [6] in measuring stability via prediction shift. However, we also measure hit ratio, the average likelihood that a top N recommender will recommend the pushed item [7]. This allows us to measure the effectiveness of the attack on the pushed item compared to all other items.

Average prediction shift is defined as follows. Let U and I be the sets of target users and items, respectively. For each user-item pair (u, i) the prediction shift denoted by $\Delta_{u,i}$, can be measured as $\Delta_{u,i} = p'_{u,i} - p_{u,i}$, where p' represents the prediction after the attack and p before. A positive value means that the attack has succeeded in making the pushed item more positively rated. The average prediction shift for an item i over all users can be computed as

$$\Delta_i = \sum_{u \in U} \Delta_{u,i} / |U|\,.$$

Similarly the average prediction shift for all items tested can be computed as

$$\bar{\Delta} = \sum_{i \in I} \Delta_i / |I|\,.$$

Note that a strong prediction shift is not a guarantee that an item will be recommended - it is possible that other items' scores are affected by an attack as well or that the item scores so low to begin with that even a significant shift does not promote it to "recommended" status. Thus, in order to measure the effectiveness of the attack on the pushed item compared to other items, we introduce the hit ratio metric. Let R_u be the set of top N recommendations for user u. For each push attack on item i, the value of a recommendation hit for user u denoted by H_{ui}, can be evaluated as 1 if $i \in R_u$; otherwise H_{ui} is evaluated to 0. We define hit ratio as the number of hits across all users in the test set divided by the number of users in the test set. The hit ratio for a pushed item i over all users in a set can then be computed as:

$$HitRatio_i = \sum_{u \in U} H_{ui} / |U|\,.$$

Likewise average hit ratio can then calculated as the sum of the hit ratio for each item i following an attack on i across all items divided by the number of items:

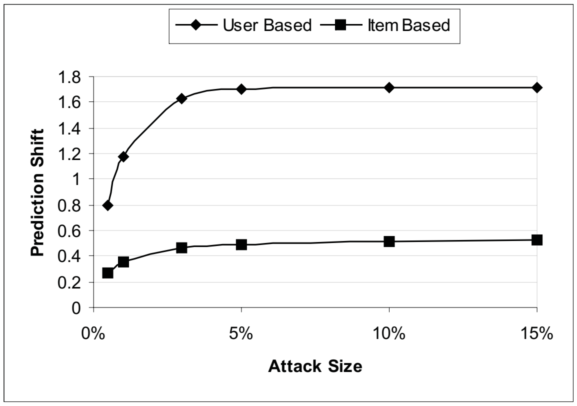


Figure 6: Average Attack in Items-Based v. User-Based Collaborative Filtering.

$$\overline{HitRatio} = \sum_{i \in I} HitRatio_i / |I| .$$

We plan to explore other metrics based on recommendation behavior, such as the bin-based techniques used in [5] and others, in our future work.

4. EXPERIMENTS AND DISCUSSION

In our experiments we have used the publicly-available MovieLens 100K dataset[1]. This dataset consists of 100,000 ratings on 1682 movies by 943 users. All ratings are integer values between one and five where one is the lowest (disliked) and five is the highest (most liked). Our data includes all the users who have rated at least 20 movies. We used a neighborhood size of 20 in the k-nearest-neighbor algorithms for both item-based and user based techniques. To perform our attack experiments, we must average over a number of different attack items, so we selected 50 movies taking care that the distribution of ratings for these movies matched the overall ratings distribution of all movies. We also generally selected a sample of 50 users as our test data, again mirroring the overall distribution of users in terms of number of movies seen and ratings provided. The results reported below represent averages over the combinations test users and test movies. We use the two metrics of prediction shift and hit ratio to measure the relative performance of various attack models. Generally, the values of these metrics are plotted against the size of the attack reported as the number of attack profiles as a percentage of the total number of profiles in the system.

Our earlier investigation [2, 1], as well as the study reported in [5], suggest that while the average and random attacks can be successful against user-based collaborative systems, they generally fall short of having a significant impact in the stability of item-based algorithms . For example, Figure 6 shows that item-based CF approach is more robust than the standard user-based algorithms in terms of the overall prediction shift on target items. Similar results were obtained when measuring the hit ratio.

In our earlier examination of user-based collaborative filtering, we examined the bandwagon attack - a lower knowledge version of the random attack and found that it was comparable in performance to the average attack without requiring as much system-specific knowledge. We repeated these experiments against the item-based algorithm with similar results, namely that despite its lower knowledge requirements, the bandwagon attack was comparable to the average attack in impact. The average attack resulted in slightly higher in both prediction shift and hit ratio measures. However, the overall impact of this attack compared to the average attack was far less successful for the item-based algorithm confirming the relative stability of the item-based algorithm over the user-based algorithm. These results are depicted in Figure 7 using 10

Should we conclude then that an item-based algorithm is a successful defense against shilling attacks, or are there specific attack models that can have a practical impact on such systems? The favorite item attack was introduced in [2] as an approach that appeared to have a theoretical advantage over previously-developed attack types when applied to the item-based collaborative filtering. Our preliminary results [1] showed that this attack model can be effective against both user-based and item-based algorithms. Here we extend those results and examine the segmented attack as a more effective and practical variation of the favorite item attack.

The favorite item attack assumes that we have knowledge of a handful of items that each user likes. Liked items are most likely to be rated - users can often predict that they will not like a particular movie and therefore avoid seeing it. Attack profiles can then be assembled that consist of these liked items and the pushed movie. Other movies are assigned low ratings. Note that a new attack must be formulated for each target user. This is not practical, of course, but if we generalize from the single user to a market niche of users with similar tastes, it becomes plausible that an attacker might construct an attack targeted only to that niche. Indeed, the attacker might have demographic and marketing data that sorts the users into market segments whose preferences might be highly predictable.

Based on this observation we introduce the segmented attack (see Figure 5), in which a set of items are selected for co-rating with the target item based on how they define a segment of users. Thus, the segmented attack targets a set of users, in contrast to individual users in the favorite item attack. Furthermore, the selection of the segment is done implicitly (without direct knowledge about the individual users within the segment) by the virtue of selecting highly rated movies with similar characteristics. To build our segmented attack, we identified a segment of users all of whom had given above average scores(4 or 5) to any three of the five movies,namely,*Alien*, *Psycho*, *The Shining*, *Jaws*, and *The Birds*.[2]

For this set of five movies, we then selected all combinations of three movies that had at least 50 users support, chose 50

[1] http://www.cs.umn.edu/research/GroupLens/data/

[2] The list was generated from on-line sources of the popular horror films: http://www.imdb.com/chart/horror and http://www.filmsite.org/afi100thrillers1.html.

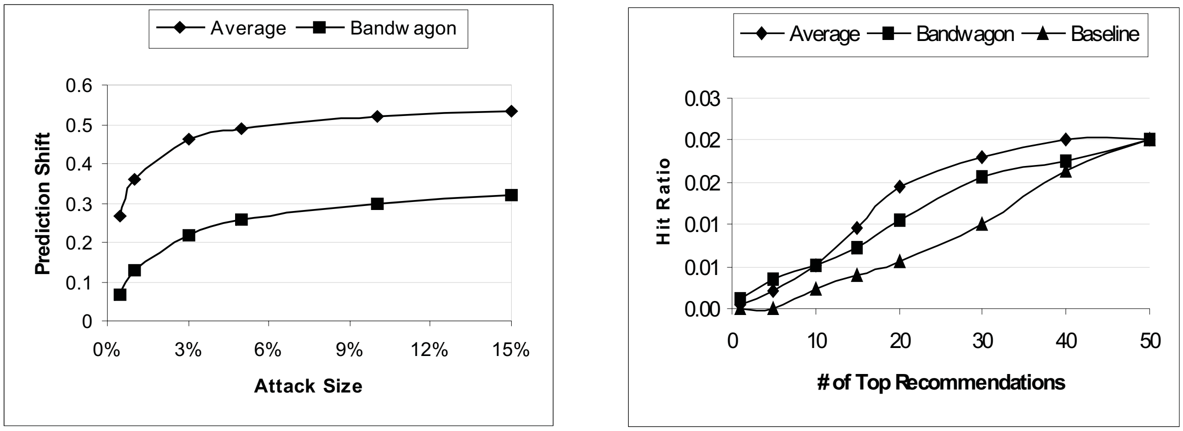


Figure 7: Comparison of Average and Bandwagon attacks in Item-Based algorithm.

of those users randomly and averaged the results. These results were also confirmed with a different segment based on Harrison Ford's movies. The power of the segmented attack is emphasized in Figure 8 in which the impact of the attack is compared within the targeted user segment and within the set of all users. Left panel in the Figure shows the comparison in terms of prediction shift and varying attack sizes, while the right panel depicts the hit ratio at 1% attack. While the segmented attack does show some impact against the system as a whole, it truly succeeds in its mission: to push the attacked movie precisely to those users defined by the segment. Indeed, in the case of in-segment users, the hit ratio is much higher than average attack.The chart also depicts the effect of hit ratio before any attack. Clearly the segmented attack has a bigger impact than any other attack we have previously examined against item-based algorithm. Our prediction shift results show that the segmented attack is more effective against in-segment users than even the more knowledge intensive average attack for the item-based collaborative algorithm.

Finally, we performed a set of experiments to determine the impact of "focus" on the segmented attack. By focus, here we mean the degree to which the user segment is characterized by its interest in a specific type of item. In this case we considered three user segments with increasing degrees of focus based on the movies they have rated highly. The first segment (focus 1) consists of all those users who have given above average rating to any one of the five horror movies as mentioned above. The second segment (focus 2) has users who have rated 4 or 5 any two of the five movies. Finally the users in the third segment (focus 3) had rated above average any three of the five movies. As the focus increases, the user segments become smaller and increasingly characterized by those who enjoy horror movies.

We performed the segmented attack against each segment, in each case taking the movie combinations described above as the selected items that were co-rated with the target item in the attack profiles. For the focus 2 and focus 1 results, the average was taken from running all combinations of the movies used in focus 3. The results of this experiment are depicted in Figure 9, showing the prediction shift and hit ratio values, respectively, across the three segments. We fixed the number of top recommendations, N, to 17 (1% of the total items). As expected, the increase in focus results in the segmented attack having a higher impact in the targeted user segment, but even a segment defined by two liked movies in common is strongly impacted by the attack.

5. CONCLUSIONS

The open and interactive nature of collaborative filtering is both a source of strength and vulnerability for recommender systems. As our research and that of others has shown, biased profile data can easily sway the recommendations of a collaborative system towards inaccurate results that serve the attacker's ends. Previous research had held out hope that item-based collaborative filtering might be relatively robust compared to the more common user-based variant. However, our research reported here shows that a fairly low-cost technique, the segmented attack, can be successfully deployed against item-based recommenders. Furthermore, the segmented attack offers pragmatic advantages for the attacker. Instead of spreading the bias due to the attacks across the whole user base, the segmented attack lets the attacker pick a focused set of users to whom an item should be pushed, effectively allowing targeted marketing of particular products to those sets of individuals judged as most likely to be influenced by the biased recommendation. We believe that hybrid recommender systems, which combine collaborative and content-based filtering, as well as model-based recommendation algorithms, may provide partial solutions the bias injection problem studied in this paper. We plan to examine the robustness of such algorithms in the context of different attack models in our future work.

6. REFERENCES

[1] R. Burke, B. Mobasher, and R. Bhaumik. Limited knowledge shilling attacks in collaborative filtering systems. In *Proceedings of the 3rd IJCAI Workshop in Intelligent Techniques for Personalization*, Edinburgh, Scotland, August 2005.

[2] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik. Identifying attack models for secure recommendation. In *Beyond Personalization: A Workshop on the Next*

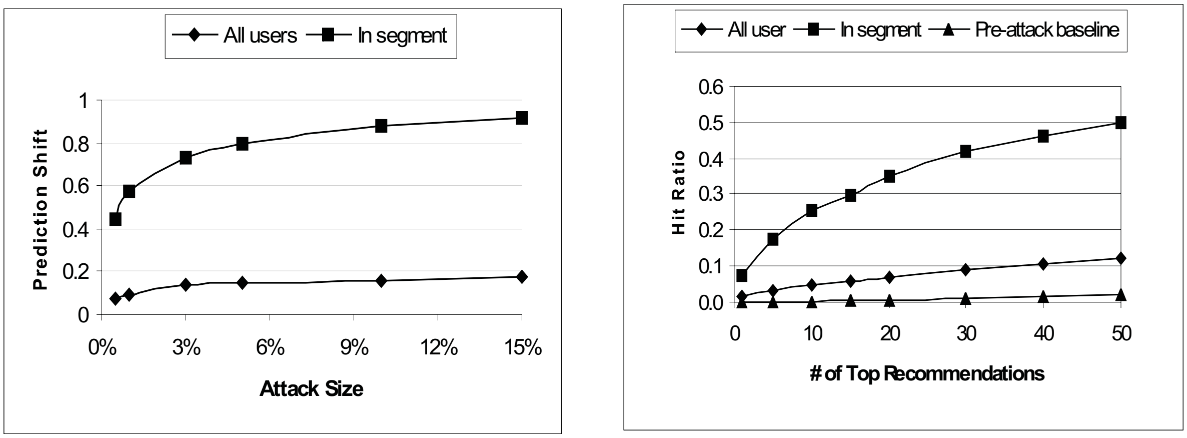


Figure 8: Segmented attack in Item-Based algorithm

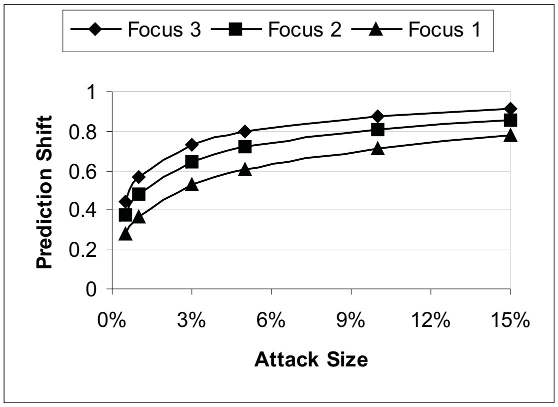


Figure 9: Analysis of in-segment focus in Segmented attack.

Generation of Recommender Systems, San Diego, California, January 2005.

[3] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, Berkeley, CA, August 1999.

[4] J.Herlocker, J. Konstan, L. G. Tervin, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[5] S. Lam and J. Reidl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th International WWW Conference*, New York, May 2004.

[6] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology*, 4(4):344–377, 2004.

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, May 2001.