Collaborative Recommendation Vulnerability To Focused Bias Injection Attacks *

Robin Burke, Bamshad Mobasher, Runa Bhaumik, Chad Williams
Center for Web Intelligence, DePaul University
School of Computer Science, Telecommunication, and Information Systems
Chicago, Illinois, USA
{rburke, mobasher, rbhaumik, cwilli43}@cs.depaul.edu

Abstract

Significant vulnerabilities have recently been identified in collaborative recommender systems. Attackers who cannot be readily distinguished from ordinary users may inject biased data in an attempt to force the system to "adapt" in a manner advantageous to them. Researchers have studied simple attack models and their impact on a system's population of users. In this paper, we examine attacks that concentrate on a targeted set of users with similar tastes, biasing the system's responses to these users. Not only are such attacks more pragmatically beneficial for the attacker (since a particular item can be pushed to those most likely to buy it), but as we show, such attacks are also highly effective against both user-based and item-based algorithms. As a result, an attacker can mount such a "segmented" attack with little knowledge of the specific system being targeted and with strong likelihood of success.

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]. Our preferred term is *profile injection attacks*, since promoting a particular product is only one way such attack might be used. In a profile injection attack, an attacker interacts with the recommender system to build within it a number of profiles associated with fictitious identities with the aim of biasing the system's output.

It is easy to see why collaborative recommendation is vulnerable to profile injection attacks. A user-based collaborative recommendation 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].

Researchers who have examined this phenomenon have concentrated on broad attack models whose profiles contains ratings across the spectrum of available objects and have measured their results by looking at how all of the users of the system are affected in the aggregate. However, it is a basic truism of marketing that the best way to increase the impact of a promotional activity is to target one's effort to those already predisposed towards one's product. In other words, it is likely that an attacker wishing to promote a particular product will be interested not in how often it is recommended to all users, but how often it is recommended to likely buyers.

If the attacker can successfully target the appropriate market segment, the relatively minor marginal utility to be gained by pushing the product to any one of the out-of-segment users may be outweighed by the increased possibility of detection that such a move entails. A rational attack strategy is therefore a segmented one: push the product to the high-probability purchasers.

This paper examines a particular attack model that we call the *segmented attack* in which the attacker concentrates on a set of items of similar content that have high visibility, the *Harry Potter* series being a good example in the book domain. It is certainly the case that these books are highly popular and widely read – it would follow that they would be rated by many users of a collaborative system. Users who enjoy these books are likely to share some characteristics: they may be children or parents who have an interest in exciting fantasy stories involving magic. These facts are general knowledge about the book domain readily available

*This research was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303.

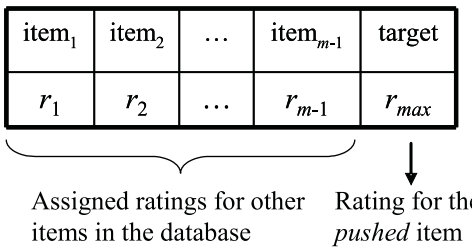

Figure 1. The general form of a push attack profile.

outside of any particular recommender system.

The segmented attack model is designed to push 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. We show that the segmented attack is both effective and practical against user-based and item-based collaborative algorithms.

The paper is organized as follows. In Section 2 we provide a general framework for profile injection attacks against collaborative systems, and we present the details of our proposed segmented attack model. Section 3 includes some background information and the specific details of the user-based and item-based recommendation algorithms used in our experiments. In Section 4 we describe our evaluation methodology, including two evaluation metrics we have used to determine the effectiveness of the segmented attack against each algorithm. We then present our experimental results, with a detailed analysis of the proposed segmented attack model, and show its effectiveness against both user-based and item-based algorithms.

2. Attack Models

A profile injection attack against a collaborative 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 attack profiles, 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. Each attack profile consists of an m-dimensional vector of ratings, where m is less than or equal to the total number of items in the system. The rating given to the pushed item is r_{max}, the maximum allowable rating value within the target recommender system. The ratings r_1 through r_{m-1} are assigned to the corresponding items according to the specific attack model. Each attack model has its own strategy for selecting the items for the attack profile and assigning ratings to them.

In the remainder of this section, we provide a detailed example that will help illustrate the vulnerability of collaborative recommendation algorithms, and will serve as a motivation for the formal description of the attack models that follow.

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.

If the system is using a standard user-based collaborative 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.[1] So, Eve's 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, Eve can make a successful attack more likely with a small amount of knowledge about the ratings distributions for some items.

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 pro-

[1] Of course, a real implementation would use more than a single neighbor for prediction, but the same principle applies with a larger number of neighbors.

	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.

files 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 successful regardless of whether the system is using an item-based or a user-based algorithm. This latter observation illustrates the motivation behind the attack model we introduce and analyze in this paper, namely the segmented attack.

2.2 The Segmented Attack

Prior work on recommender system stability has examined primarily three attacks. The sampling attack from [6] is primarily of theoretical interest as it requires the attacker to have access to the ratings database itself. The random attack [5] forms profiles by associating a positive rating for the target item with random values for the other items. The average attack [5] assumes that the attacker knows the average rating for each item in the database and assigns values randomly distributed around this average, except for the target item. This attack has been found to be effective against user-based collaborative recommendation algorithms, but less so against item-based recommendation.

Each of these prior attack models makes the implicit assumption that the attacker is interested in promoting the pushed item to every user in the system. However, suppose that Eve in our previous example had written a fantasy book for children. She would no doubt prefer that her book be recommended to buyers who had expressed an interest in this genre, for example buyers of *Harry Potter* books, rather than buyers of books on Java programming or motorcycle repair. Eve would rightly expect that the "fantasy book buyer" segment of the market would be more likely to respond to a recommendation for her book than others.

We can frame this intuition as a question of utility. We assume that the attacker has a particular item i that she wants recommended more highly because she has a personal stake in the success of this product. The attacker receives some positive utility or profit p_i each time i is purchased. In biasing the recommender system, the attacker hopes to increase the probability that purchases will happen, but of course not every user to whom a recommendation is made will actually purchase. Let us denote the event that a recommendation of product i is made to a user u, by $R_{u,i}$ and the event that a user buys an item by $B_{u,i}$. The probability that a user will purchase i if it is recommended we can describe as a conditional probability: $P(B_{u,i}|R_{u,i})$. Over all users U that visit the system over some time period, the expected profit would be

$$P = \sum_{u \in U} p_i * P(R_{u,i}) * P(B_{u,i}|R_{u,i})$$

The attacker of a recommender system hopes to increase her profit by increasing $P(R_{u,i})$, the probability that the system will recommend the item to a given user.

However, preferences for most consumer items are not uniformly distributed over the population of buyers. For many products, there will be users (like a "Harry Potter" buyers) who who would be susceptible to following a recommendation for a related item (another fantasy book for children) and others who would not. In other words, there will be some segment of users S that are distinguished from the rest of the user population $N = U - S$, by being likely recommendation followers:

$$\forall s \in S, \forall n \in N, P(B_{s,i}|R_{s,i}) \gg P(B_{n,i}|R_{n,i}))$$

Let us consider an extreme case of a niche market in which $P(B_{n,i}|R_{n,i})$ is zero. The only customers worth recommending to are those in the segment S. Everyone else

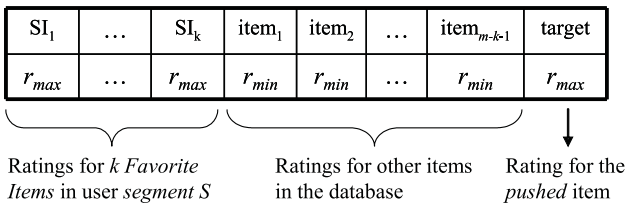

Figure 3. General form of the Segmented Attack.

will ignore the recommendation. It is in the attacker's interest to make sure that the attacker item is recommended to the segment users; it does not matter what happens to the rest of the population. The attacker will be only interested in manipulating the quantity $P(R_{s,i})$. In other words, the quantity that matters to an attacker may not be the overall impact of an attack, but rather its impact on a segment of the market distinguished as likely buyers. This may even be true if $P(B_{n,i}|R_{n,i}) > 0$ because these out-of-segment buyers contribute relatively little to the expected utility compared to the in-segment ones.

Obviously, the maximum P is realized when every single user gets the pushed item as a recommendation. That is, when $P(R_{u,i}) = 1$. That may not be a realistic goal. Only a very large attack (perhaps with a number of biased profiles equal to or greater than the size of the original profile database) would be able to ensure such an effect, and such an attack would be likely to be noticed by a site's operators. In addition, the ubiquity of the pushed item would be noticed by users for whom it is not a good match: buyers of motorcycle repair books suddenly getting recommendations for children's fantasy titles might complain, and the complaints would form a detectable pattern. This increased risk of detection is a cost associated with large attack sizes. Therefore, it is rational for the attacker to focus solely on the in-segment users to the extent that this is possible.

We define a segment as a set of users with shared strong favorable preferences for a set of segment items (such as "Harry Potter" books in our example.) Let SI be the set of items that define a target segment. To target the users in the segment, we construct profiles with high ratings for the items in the set SI and low ratings for other items. These profiles will match users who also have a strong preference for the items in SI. See Figure 3. An attacker like Eve only needs to identify books that are similar to the one she wants to push and relatively popular in order to generate the attack.

3. Recommendation Algorithms

This paper reports on results for two of the most commonly-used collaborative algorithms: user-based and item-based collaborative recommendation using nearest-neighbor techniques [3, 7]. In each case, the algorithm assumes there is a single user / item pair for which a prediction is sought – in our experiments this is generally the pushed item, since we are primarily interested in the impact that attacks have on this item.

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:

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

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.

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

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.

Item-based collaborative filtering works by comparing items based on their pattern of ratings across users. Again, a nearest-neighbor approach can be used, but here a more common approach is the adjusted cosine similarity measure introduced by [7]. The adjusted cosine similarity formula is given by:

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

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. In this measure, all user profiles are normalized by subtracting the user's mean rating. When items are compared, the ratings given by each user to that item are combined in a vector and the similarity between them is calculated as the vector cosine.

After computing the similarity between items we select a set of k most similar items to the target item and generate a predicted value:

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

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 adjusted cosine similarity between items i and j. The user's own ratings of similar items are used to extrapolate the prediction for the target item. We consider a neighborhood of size 20 and ignore items with negative similarity.

4. Experiments

In our experiments we use the publicly-available MovieLens 100K dataset[2]. 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.

4.1 Methodology

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. The metrics of *stability* and *robustness* were introduced in [6].

Our interest is along the lines of stability: how the attack changes the system's ratings for the pushed item, but more generally we are interested in measuring the effectiveness of an attack - the "win" for the attacker. The desired outcome for the attacker in a "push" attack is 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*, the change in predicted rating for the target item after the attack. However, we also measure *hit ratio*, the average likelihood that a top N recommender will recommend the pushed item [7].

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 can be computed by averaging $\Delta_{u,i}$ over all users, and an overall average can be generated by picking a number of different items to attack and averaging over them. We chose 50 movies at random from the MovieLens data, being careful that this set of target items mirrored the distribution of the data as a whole.

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 target item scores so low to begin with that even a significant shift does not promote it to "recommended" status. It is a good rough indicator of the success of an attack, but it does not get at our notion of a "win": increased probability of recommendation. In order to measure the benefit of the attack from the attacker's point of view, we use the notion of the hit ratio. The idea is to establish a window of size N at the top of the recommendation list. We count a success – a hit – if the pushed movie shows up in this window.

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 $i \in R_u$; 0, otherwise. 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, computed as: $HitRatio_i = \sum_{u \in U} H_{ui} / |U|$. The average hit ratio can then be calculated as the sum of the hit ratios for attacks on each item i across all items divided by the number of items.

For the segmented attack, we investigated two market segments: one defined by Harrison Ford's action movies and one by popular horror films. Recall that the segmented attack is constructed by identifying a set SI of segment items and the attacked users are the ones who have rated those items highly. In the Harrison Ford segment, the movies were *Star Wars*, *Return of the Jedi*, *Indiana Jones and the Last Crusade*, and *Raiders of the Lost Ark*. In the Horror segment, the movies were *Alien*, *Psycho*, *The Shining*, *Jaws*, and *The Birds*.[3]

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

[3] This list was generated from on-sources of the pop-

For the Harrison Ford segment, we chose those users who had given top rating (5) to all four movies. From this set, we chose 50 users at random. For the Horror movie segment, we chose those users who had given above average scores (4 or 5) to any three of the five movies. For this set of five movies, we selected all combinations of three movies that had at least 50 users support, chose 50 of those users randomly and averaged the results.

For all the attacks, we generated a number of attack profiles and inserted them into the system database and then generated predictions. We measure "size of attack" as a percentage of the pre-attack user count. There are approximately 1000 users in the database, so an attack size of 1% corresponds to 10 attack profiles added to the system.

4.2 Experimental Results

If we evaluate the segmented attack based on its average impact on all users, there is nothing remarkable. The attack has an effect but does not approach the numbers reached by the average attack, the most effective attack we had previously studied [1]. However, we must recall our market segment assumption: namely, that recommendations made to in-segment users are much more useful to the attacker than recommendations to other users. Our focus must therefore be with the "in-segment" users, those users who have rated the segment movies highly and presumably are desirable customers for pushed items that are similar: an attacker using the Harrison Ford segment might be interested in pushing a new movie featuring the star in an action role.

The intuition behind the segmented attack is borne out in Figure 4. The figure shows prediction shift results for the Harrison Ford segment, comparing all users with in-segment users. The in-segment prediction shift is slightly stronger for the segmented attack than the average attack. Note also that the segmented attack requires considerably less knowledge of the ratings distribution in the system than the average attack requires. ([1] discusses the question of limited knowledge attacks in greater detail.)

The hit ratio results are shown in Figure 5 for a 1% attack at different values of N. These results show that even an attack as small as 1% on the user based algorithm can have a major impact on the hit ratio.[4] It is also interesting that although the overall user base is not affected as much as the in-segment users, the shift is still very large, more than a whole point on the rating scale with a 1% attack and with the target movie showing up in the top five more than 40% of the time. The most likely reason for this is that some of the movies in this segment (such as *Star Wars* and *Raiders of the Lost Ark*) were rated highly by a majority of users in the database. Essentially, there is not that much difference between the in-segment users and the rest of the user base with respect to these movies, no doubt having to do with the characteristics of the population using the MovieLens system at the time the ratings were collected.

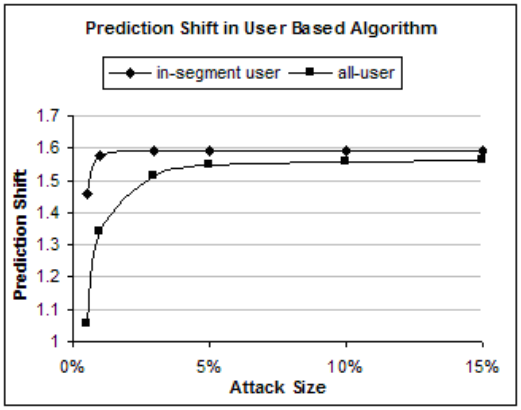

Figure 4. Prediction Shift results for the Harrison Ford segment. User-based algorithm.

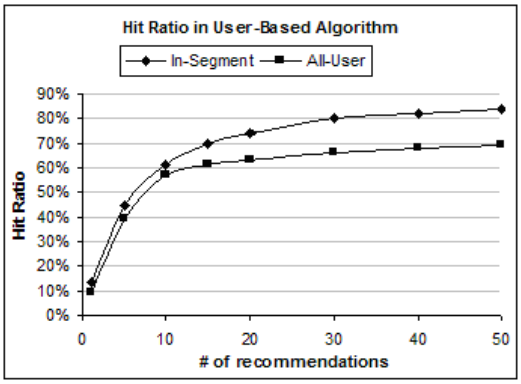

Figure 5. Hit Ratio results for the Harrison Ford segment. User-based algorithm.

The benefit of the segmented attack is considerably more striking in the item-based case shown in Figures 6 and 7. Lam and Reidl concluded, based on their results with the random and average attacks, that item-based algorithms were more robust than user-based ones [5]. However, as the figures show, the segmented attack works well against the item-based algorithm. The reason has to do with profile construction. Since the segmented attack assigns maximum ratings to both the segment items SI and the target item, the similarity between these items and the target item is in-

ular horror films: http://www.imdb.com/chart/horror and http://www.filmsite.org/afi100thrillers1.html.

[4]The hit ratio prior to the attack is very small, about 1% at N = 10 and less than 5% even with an N of 50.

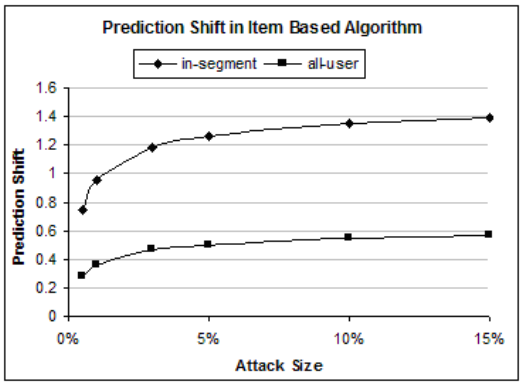

Figure 6. Prediction Shift results for the Harrison Ford segment. Item-based algorithm.

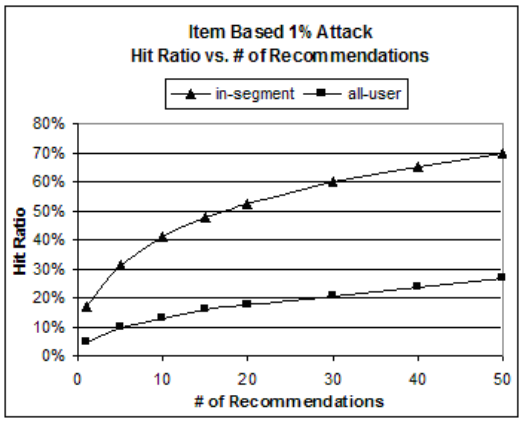

Figure 7. Hit Ratio results for the Harrison Ford segment. Item-based algorithm.

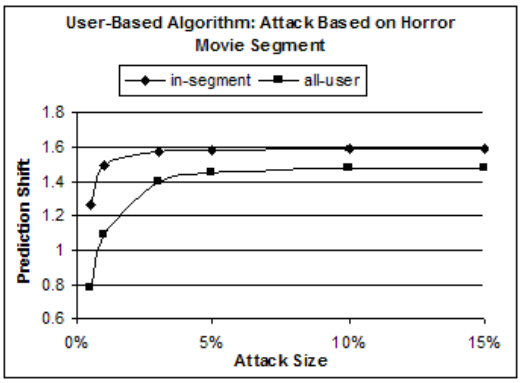

Figure 8. Prediction Shift results for the Horror Movie segment. User-Based algorithm.

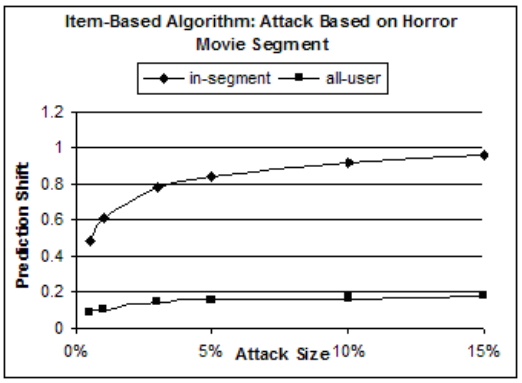

Figure 9. Prediction Shift results for the Horror Movie segment. Item-based algorithm.

creased. The low ratings given to the other items makes them more distant. If the segment items are in the algorithm's prediction neighborhood for the target item, they will boost the recommendation scores since these are items that an in-segment user will have rated highly.

In the case of the Horror movie segment, the movies were selected from on-line sources as the best movies of their type but none of them are as broadly popular as *Star Wars*. So, these movies represent more of a market niche. Figure 8 shows a similar result to that seen with prediction shift for the Harrison Ford segments against the user-based algorithm. Figure 9 indicates the focused manner in which this attack homes in on its target audience when the item-based algorithm is attacked. The general population is barely effected by the injected profiles, but there is a sizable prediction shift for in-segment users. The hit ratio results for this user segment are depicted in Figures 10 and 11 and are similar to those already seen.

These results also point out an interesting difference between the user-based and item-based algorithms. While, is both cases, the attack has a dramatic impact on the in-segment users, the overall impact of the segmented attack on the whole user group is more pronounced in the case of user-based algorithm.

Another way in which the item-based algorithm shows robustness is with respect to *profile size*. In the segmented attack, the items that are not in the SI set (see Figure 3) are given low values. In our initial experiments with the attack, all such movies were used in the attack profile. We define this as a profile size of 100%. However, this means that each attack profile must be very large, perhaps unrealistically so. We experimented with decreasing the number

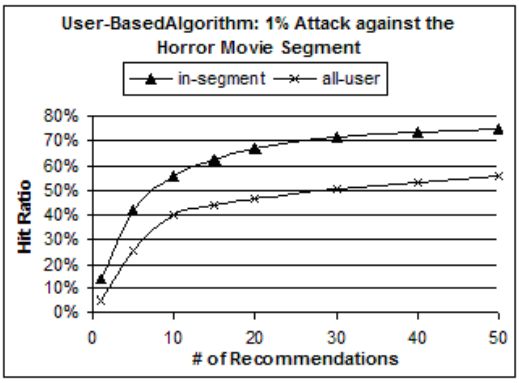

Figure 10. Hit Ratio results for the Horror Movie segment. User-Based algorithm.

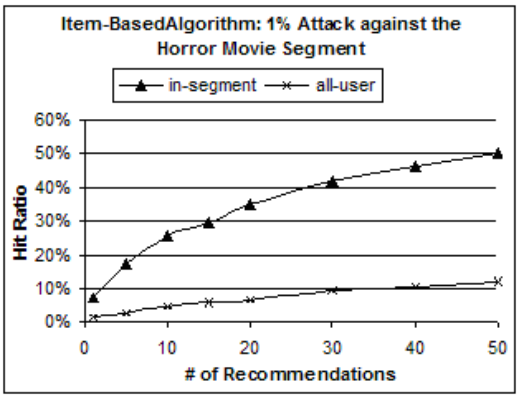

Figure 11. Hit Ratio results for the Horror Movie segment. Item-based algorithm.

of "non-favorite" items rated in each attack profile, leaving these items unrated. Interestingly, there is a peak at a low value (about 3%, or about 50 movies) when the user-based algorithm is attacked. It is this 3% profile version of the attack that was used in the experimental results shown above. As Figure 12 shows, the item-based algorithm has no such peak: the prediction shift increases monotonically for larger profile sizes. Item-based recommendation would therefore appear to have an additional advantage over user-based — an attacker must build larger profiles to be successful.

5. Conclusions

Previous research has examined profile injection attacks against recommender systems that are broad in their construction and impact. Of these, the average attack has been found to be most effective. From a cost-benefit point of

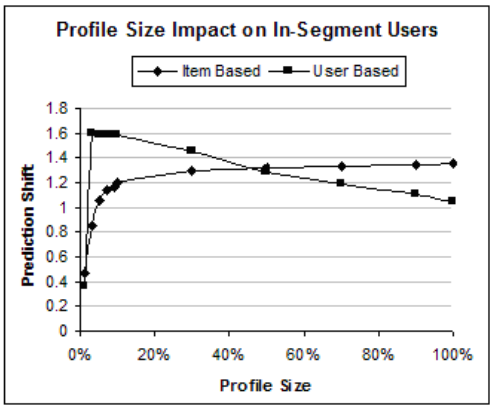

Figure 12. Comparing item-based and user-based algorithms at different profile sizes.

view, however, such attacks are sub-optimal: they require a significant degree of system-specific knowledge to mount, and they push items to users who may not be likely purchasers. In addition, they are not effective against item-based implementations.

In this paper, we introduce the segmented attack, a profile injection attack that associates the pushed item with a small number of popular items of similar type. As our results show, the attack does well at ensuring that the pushed item will be recommended to those users that are its target market. It is effective against item-based recommendation algorithms to a degree that broader attacks are not, and has no requirement for system-specific ratings distribution data.

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 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.