

Analysis and Detection of Segment-Focused Attacks Against Collaborative Recommendation^{*}

Bamshad Mobasher, Robin Burke, Chad Williams, and Runa Bhaumik

Center for Web Intelligence
School of Computer Science, Telecommunication and Information Systems
DePaul University, Chicago, Illinois
(mobasher,rburke,cwilli43,rbhaumik)@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 can easily introduce biased data in an attempt to force the system to “adapt” in a manner advantageous to them. Our research in secure personalization is examining a range of attack models, from the simple to the complex, and a variety of recommendation techniques. In this chapter, we explore an attack model that focuses on a subset of users with similar tastes and show that such an attack can be highly successful against both user-based and item-based collaborative filtering. We also introduce a detection model that can significantly decrease the impact of this attack.

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 [1–4], but we call *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 filtering is vulnerable to profile injection 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 [3] and [4]. This vulnerability of collaborative filtering has been the focus of some recent studies directed at better understanding the weakness of these recommendation algorithms. These explorations have shown that high-knowledge [2]

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

attack models can be very effective at attacking user-based collaborative filtering, but the item-based collaborative filtering algorithm [5] appears to be more robust. These findings have been further supported by the lack of success of reduced knowledge attacks against the item-based algorithm.

Researchers who have examined this phenomenon have concentrated on broad attack models whose profiles contain 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 toward one's product. In other words, it is more 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 the particular market segment that is likely to already have a propensity to purchase it.

We explore a new style of attack model that shifts the focus from trying to impact the complete user set to instead targeting a segment of profiles with specific interests. We call such an attack a *Segment Focused attack* (or simply a *segment attack*). As we show, an attack based on this approach can not only be highly effective against the user-based algorithm, but can also have a significant impact on the item-based algorithm, which is generally considered more robust. Furthermore this attack model demonstrates both algorithms are vulnerable to attacks requiring only a limited amount of system knowledge. Our experiments compare the segment attacks with the standard attack models previously studied, and demonstrate the segment attack model's effectiveness against item-based collaborative filtering algorithm.

The paper concludes with an examination of how a system might insulate itself from such an attack through detection and response. Recently a few researchers have focused on detecting and preventing shilling attacks. Chirita et al. [6] proposed several metrics for analyzing rating patterns of malicious users and evaluate their potential for detecting such shilling attacks. Su et al. [7] developed a spreading similarity algorithm in order to detect groups of very similar shilling attackers which is applied to a simplified attack scenario. O'Mahony et al. [8] developed several techniques to defend against the attacks proposed in [3] [4]. We show empirically that previous detection algorithms that have been effective against more traditional attacks are not necessarily effective against the segment attack. In response, we introduce a new classification-based detection algorithm, called the *Segment Model Detection* which uses a set of detection attributes for classifying attack profiles. We show that the proposed detection model when combined with either collaborative technique can greatly increase the system's robustness in the face of a segment attack.

The paper is organized as follows. In Section 2 we provide a general formal framework for profile injection attacks against collaborative systems, and we present the details of our proposed segment 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 use 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 that it can be effective against both user-based and item-based algorithms. In Section 5 we provide a brief introduction to attack detection and response and work that has been done to address this need. Following this, we introduce a model based technique for detecting and responding to the segment attack. In Section 5.3 we present experimental results of our model-based approach compared with the Chirita algorithm.

2 Attack Models

We have considered attacks where the attacker’s aim is to introduce a bias in the recommender system by injecting fake user ratings. We call such attacks *profile injection attacks*. Our goal is to identify different types of profile injection attacks, and to study their characteristics and their impact on common collaborative filtering recommendation algorithms. 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 or less highly. In this paper, our focus is on “shilling” attacks in which the aim of the attacker is to promote the target item. Hence, we call such attacks, *push* attacks. 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 consists of an m -dimensional vector of ratings, where m is the total number of items in the system. The rating given to the pushed item, i_t , is r_t . Generally, in a push attack, $r_t = r_{max}$, i.e., the maximum rating on the rating scale. On the other hand, in a *nuke* attack, in which the goal is to reduce the likelihood of the target item being recommended, $r_t = r_{min}$, where r_{min} is the minimum allowable rating.

We begin by presenting a formal framework for characterizing attack models and attack profiles. We then turn our attention to several specific attack models that we have introduced or studied in this work and discuss their properties. In Section 4, we present our detailed experimental results corresponding to these attack models.

2.1 Framework for Characterizing Profile Injection Attacks

Let I be a set of items, U a set of users, R a set of rating values, r_{max} is the maximum rating in R , and r_{min} is the minimum rating in R , and $UP = \{up_1, up_2, \dots, up_d\}$ a set of user profiles, where each $up_i = \{\langle i, r \rangle \mid i \in I, r \in R \cup \{\text{null}\}\}$, with **null** representing a missing or undefined rating.

An *attack model* partitions the set of items, I , into 4 groups i_t , I_S , I_F , and I_\emptyset , such that i_t is the *target item*; I_S is a set of *selected items*; I_F is a set of randomly selected *filler items*; and $I_\emptyset = I - (I_S \uplus I_F \uplus \{i_t\})$ is the set of *unrated items*.

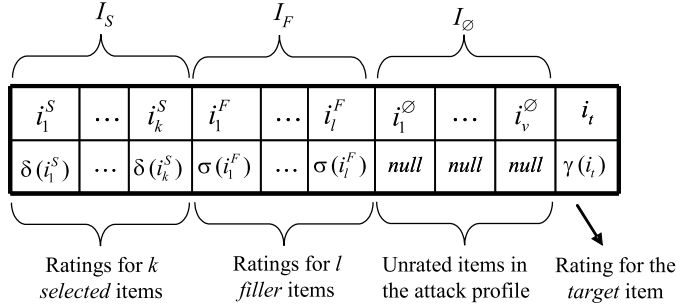


Fig. 1. The general form of an attack profile in a profile injection attack.

The set of parameters that define the partitioning and assignment of ratings for each partition are specific to the particular attack model. The functions δ and σ assign ratings to the elements I_S and I_F , respectively, while γ assigns a rating to the target item, i_t . These functions, and the manner in which I is partitioned, depend on the particular attack model, as described below.

The generic form of an attack profile, as described above, is depicted pictorially in Figure 1. Generally speaking, specific attack models vary based on the parameters or methods used to identify the *selected items*, the proportion of the remaining items that are used as *filler items*, and the way that specific ratings are assigned to each of these sets of items and to the target item. In most cases, the set of selected items, I_S , is either empty or represents a small group of items that have been selected because of their association with the target item (or a targeted segment of users). These items may be selected according to a number of factors that may include the distribution of rating values among items or users, the likelihood that a particular item is highly or frequently rated, or the expected characteristics associated with a particular segment of users. On the other hand, the set of filler items, I_F , represent a group of randomly selected items in the database which are assigned ratings within the attack profile.

We next focus our attention on a number of specific attack models and discuss some of their characteristics.

2.2 Basic Attack Models

Two attack models, introduced originally in Lam and Reidl [3] are the *random* and *average* attack models. Both of these attack models involve two groups of items, the target item i_t and a set of filler items I_F where all items in I_F are assigned ratings based on the same amount of system knowledge. In [3], these models assumed that the filler items consisted of all items in the databases (except the target items), potentially requiring substantial knowledge and effort on the part of the attacker. We introduce a more general form of these attacks that allows for varying degrees of knowledge about the items in the system. We have also introduced a more sophisticated attack model, *Bandwagon attack* [2],

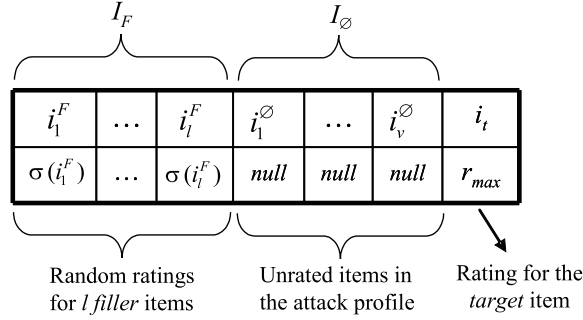


Fig. 2. An attack profile based on the random or the average attack model.

that is designed to be more practical in terms of the knowledge or the effort involved by the attacker.

Both random and average attack profiles consist of ratings assigned to the filler items, as described above, and a pre-specified rating assigned to the target item. In the random attack the assigned ratings are based on the overall distribution of user ratings in the database, while in the average attack the rating for each filler item is computed based on its average rating for that item. Although we generally treat this attack model as a push attack, it can also be used as a nuke attack by changing the assigned rating for the target item from r_{max} to r_{min} . In both of these attack models, the set of selected items is empty. More formally, the average and random attack models have the following general characteristics:

- $I_S = \emptyset$;
- I_F is a set of randomly chosen filler items drawn from $I - \{i_t\}$, where the ratio of filler items $|I_F|/|I - \{i_t\}|$ is a pre-specified parameter.
- $\forall i \in I_F, \sigma(i) \sim \mathcal{N}(\mu, s)$, where $\sigma(i)$ is the rating given to item i . For average attack, $\mu = \mu_i$ and $s = s_i$, where μ_i and s_i are the mean and standard deviation of ratings of item i across all users, i.e., the rating value for each item $i \in I_F$ is drawn from a normal distribution around the mean rating for i . In the random attack $\mu = \mu_I$ and $s = s_I$, where μ_I and s_I are the mean and standard deviation of ratings for all items in I , i.e., the rating value for each item $i \in I_F$ is drawn from a normal distribution around the mean rating value across all items and users in the database;
- $\gamma(i_t) = r_{max}$, where r_{max} is the maximum rating value in R .

It should also be noted that the only difference between the average attack and the random attack is in the manner in which ratings are assigned to the filler items in the profile. Figure 2 depicts the general form for both the random and the average attacks.

The knowledge required to mount random 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). However, as

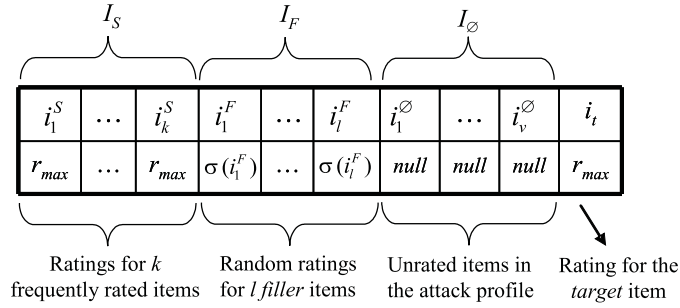


Fig. 3. A *Bandwagon* attack profile.

Lam and Reidl [3] show, and our results confirm [2], the attack is not particularly effective.

Average attack which is more powerful uses the individual mean for each item rather than the global mean (except for the pushed item). In addition to the effort involved in producing the ratings, the average attack also has a considerable knowledge cost of order $|I_F|$ (the number of filler items in the attack profile). Our experiments, however, have shown that, in the case of user-based collaborative filtering algorithms, the average attack can be just as successful even when using a small filler item set, i.e., by assigning the average ratings to only a small subset of items in the database. Thus, the knowledge requirements for this attack can be substantially reduced [2]. This attack model, however, is not, in general, effective against an item-based collaborative algorithm, as we will show in Section 4.

The bandwagon attack [2] is based on the random attack, except that an additional group of items is selected to increase effectiveness with a minimal amount of additional knowledge. The goal of the bandwagon attack is to associate the attacked item with a small number of frequently rated items. 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. This attack can be considered to have low knowledge cost since 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.

More formally, the *Bandwagon attack model* is an attack model, with the following characteristics:

- I_S is a set of selected items, such that for each $i \in I_S$, $|\{(i, r) \in up_j \mid up_j \in UP, r \neq \text{null}\}| \geq c$, where c is a pre-specified threshold constant (i.e., the total number of non-null ratings for all elements of I_S exceed a pre-specified constant).

- $\forall i \in I_S, \delta(i) = r_{max}$, where r_{max} is the maximum rating value in R .
- I_F is a set of randomly chosen filler items drawn from $I - \{i_t\}$, where the ratio of filler items is of size $|I_F|/|I - \{i_t\}|$ and is a pre-specified parameter;
- $\forall i \in I_F, \sigma(i) \sim \mathcal{N}(\mu_I, s_I)$, μ_I and s_I are the mean and standard deviation of ratings for all items in I , i.e., the rating value for each item $i \in I_F$ is drawn from a normal distribution around the mean rating value across the whole database;
- $\gamma(i_t) = r_{max}$.

Figure 3 depicts a typical attack profile for the bandwagon attack. Items i_1^S through i_k^S in I_S are selected 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, i_t . The ratings for the filler items i_1^F through i_l^F in I_F 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 Burke et al. [2] that, despite the reduced knowledge, the bandwagon attack can still very successful against user based collaborative filtering. However, as we show below, none of these attacks are particularly effective against the item based algorithm.

2.3 The Segment Attack Model

Previous work [3] concluded that item-based algorithms were more robust than user-based ones and the average attack has been found to be most effective. From a cost-benefit point of 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. To address this, we introduce the *segment attack* model as a reduced-knowledge push attack specifically designed for the item-based algorithms [9].

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. The segment attack model is designed to push an item to a targeted group of users with known or easily predicted preferences. For example, 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. In addition, it would be to the benefit of the attacker to reduce the impact to unlikely buyers since a broad attack will be easier to detect.

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. 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

$$P = \sum_{u \in U} p_i * P(R_{u,i}) * P(B_{u,i}|R_{u,i}) \quad (1)$$

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 ‘‘Harry Potter’’ buyers) 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:

$$\forall s \in S, \forall n \in N, P(B_{s,i}|R_{s,i}) \gg P(B_{n,i}|R_{n,i}) \quad (2)$$

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 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.

More formally, the *segment attack model* is an attack model, with the following characteristics:

- I_S is a set of selected items the attacker has chosen to define the segment, and all items in I_S are given the rating r_{max} (i.e., $\delta(i_j^S) = r_{max}$).
- I_F is a set of randomly chosen filler items drawn from $I - \{I_S \uplus i_t\}$, where the ratio of filler items $|I_F|/|I - \{i_t\}|$ is a pre-specified parameter, and all items in I_F are given the rating r_{min} (i.e., $\delta(i_j^F) = r_{min}$);
- $\gamma(i_t) = r_{max}$.

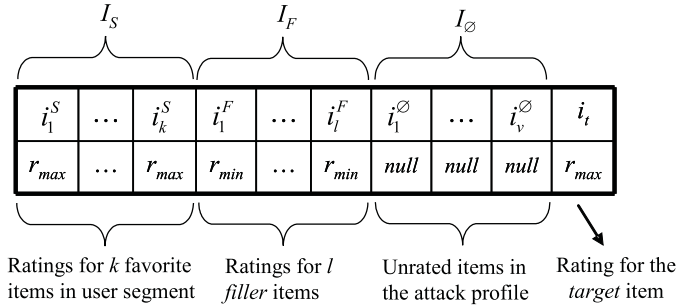


Fig. 4. A *Segment* attack profile.

The target group of users (segment) in the segment attack model can then be defined as the set $U_S = \{UP_1^S \dots UP_t^S\}$ of user profiles in the database such that: $\forall UP_j^S \in U_S, \forall i \in I_S, Rating(UP_j^S, i) \geq r_c$, where $Rating(UP_j^S, i)$ is the rating associated with item i in the profile UP_j^S , and r_c is a pre-specified minimum rating threshold.

Figure 4 depicts a typical attack profile based on the segment attack model. The selected segment items, i_1^S through i_k^S in I_S represent the items that are (likely to be) favored by the targeted segment of users, like the Harry Potter books in the above example. These items are assigned the maximum rating value together with the target item. To provide the maximum impact on the item-based CF algorithm, the minimum rating was given to the filler items, thus maximizing the variations of item similarities used in the item-based algorithm.

The detailed experimental results for this attack model are presented in Section 4. The result show that this attack model is quite effective against both item-based and user-based collaborative filtering.

3 Recommendation Algorithms and Evaluation Metrics

We have focused on the most commonly-used algorithms for collaborative filtering, namely user-based and item-based. Previous work had suggested that item-based collaborative filtering might provide significant robustness compared to the user-based algorithm, but, as this paper shows, the item-based algorithm also is still vulnerable in the face of some of the attacks we introduced in the previous section. In the rest of this section we provide the details of the standard CF algorithms we have used in our experiments.

3.1 User-Based Collaborative Filtering

The standard collaborative filtering algorithm is based on user-to-user similarity [10]. This k NN algorithm operates by selecting the k most similar users to the target user, and formulates a prediction by combining the preferences of these users. k NN 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:

$$sim_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u) * (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} * \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}} \quad (3)$$

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 .

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in V} sim_{u,v} * (r_{v,i} - \bar{r}_v)}{\sum_{v \in V} |sim_{u,v}|} \quad (4)$$

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. Note also that if the denominator of the above equation is zero, our algorithm replaces the prediction by average rating of that user u .

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 k NN algorithm attempts to find k similar items that have a similar pattern of ratings across different users.

For our purpose we have adopted the adjusted cosine similarity measure introduced by Sarwar et al. [5]. The adjusted cosine similarity formula is given by:

$$sim_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) * (r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2} * \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2}} \quad (5)$$

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:

$$p_{u,i} = \frac{\sum_{j \in J} r_{u,j} * sim_{i,j}}{\sum_{j \in J} sim_{i,j}} \quad (6)$$

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. As in the case of user-based algorithm, if the denominator of the above equation is zero, our algorithm replaces the prediction by average rating of that user u .

3.3 Evaluation Metrics

There has been considerable research in the area of recommender systems evaluation [11]. 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 O’Mahony et al. [4] 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 O’Mahony et al. [4] 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 [5]. 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:

$$\Delta_i = \sum_{u \in U} \Delta_{u,i} / |U|. \quad (7)$$

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

$$\bar{\Delta} = \sum_{i \in I} \Delta_i / |I|. \quad (8)$$

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:

$$HitRatio_i = \sum_{u \in U} H_{ui} / |U|. \quad (9)$$

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:

$$\overline{HitRatio} = \sum_{i \in I} HitRatio_i / |I|. \quad (10)$$

4 Experimental Results

In our experiments we have used the publicly-available Movie-Lens 100K dataset¹. 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.

In all experiments, we used a neighborhood size of 20 in the k -nearest-neighbor algorithms for user-based and item-based systems. To perform our attack experiments, we must average over a number of different attack items, so we selected 50 movies from a wide range of ratings. 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 of test users and test movies. We use the metrics of prediction shift and hit ratio, as described earlier, to measure the relative performance of various attack models. Generally, the

¹ <http://www.cs.umn.edu/research/GroupLens/data/>

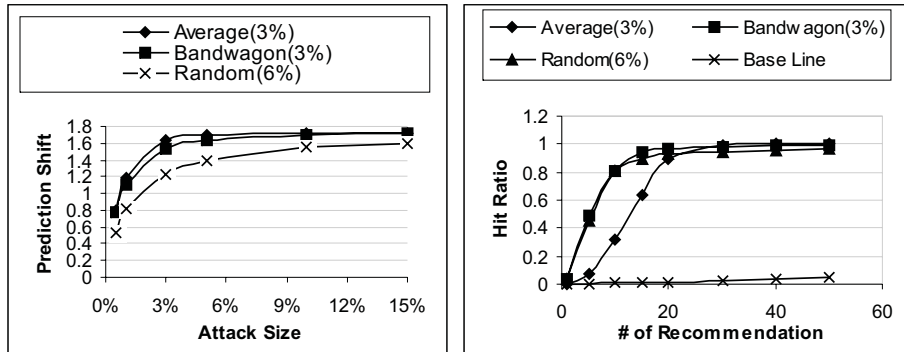


Fig. 5. Comparison of attacks against user-based algorithm, prediction shift (left) and hit ratio(right). The baseline in the right panel indicates the hit ratio results without any attack.

values of these metrics are plotted against the size of the attack reported as a percentage of the total number of profiles in the system.

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.

In the results below, we present the vulnerabilities of both user-based and item-based collaborative filtering against push attacks.

4.1 Attacks Against User-Based Collaborative Filtering

Figure 5 shows the results of a comparative experiment examining three algorithms at different attack sizes. The algorithms include the average attack (3% filler sizes), the bandwagon attack (using 1 frequently rated item and 3% filler size), and the random attack (6% filler size). We see that even without system-specific data an attack like the bandwagon attack can be successful at higher attack sizes. The best performance is achieved with profiles with relatively small filler sizes. The average attack is obviously quite powerful - recall that the rating scale in this domain is 1-5 with an average of 3.6, so a rating shift of 1.5 is enough to lift an average-rated movie to the top of the scale. On the other hand, the bandwagon attack is able to come very close to the average attack in performance, even with minimal knowledge requirements. All that is necessary for an attacker is to identify a few items that are likely to be rated by many users.

4.2 Attacks Against Item-Based Collaborative Filtering

Our earlier investigation [1], as well as the study reported in Lam and Reidl [3], suggest that while the average and random attacks can be successful against

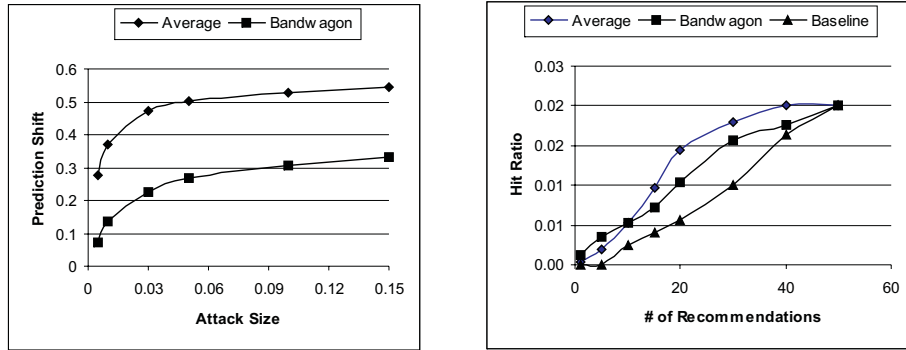


Fig. 6. Prediction Shift(left) and HitRatio(right) results for Average and Bandwagon Attack against Item-Based Collaborative Filtering. The Baseline in the right panel indicates the hit ratio results without any attack

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. Note the difference in the y-axis scales. Similar results are seen with the hit ratio metric.

Should we conclude then that an item-based algorithm is a successful defense against profile injection attacks, or are there specific attack models that can have a practical impact on such systems? Our experiments with the segment attack (see Figure 4) suggest that the answer to the later question is yes.

Recall that in the segment attack a set of items are selected for co-rating with the target item based on how they define a segment of users. 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. The bandwagon attack uses general knowledge about the popularity of items. The segmented attack is based on a different type of knowledge. It uses domain knowledge about the similarity of items in order to target users that prefer items of a particular type. 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. 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.

To build our segmented attack profiles, we identified the segment of users as all users who had given above average scores (4 or 5) to any three of the five most popular horror movies, namely, *Alien*, *Psycho*, *The Shining*, *Jaws*, and *The*

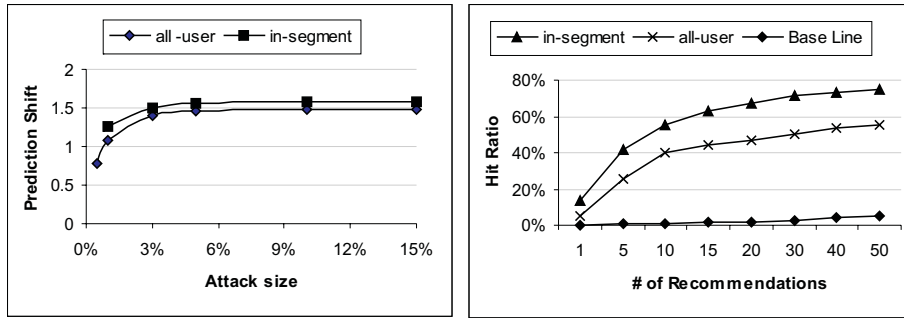


Fig. 7. Prediction shift and hit ratio results for the Horror Movie Segment in user-based algorithm.

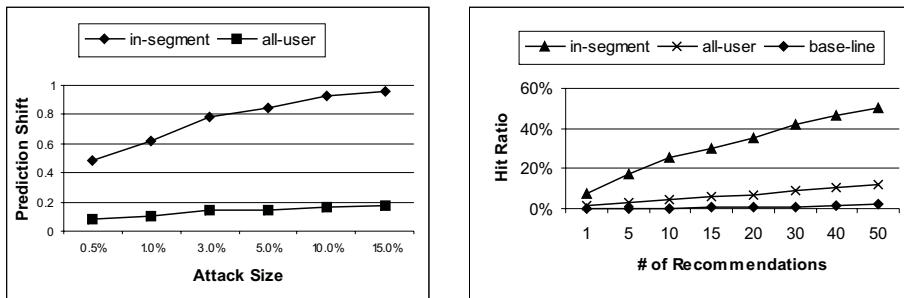


Fig. 8. Prediction shift and hit ratio results for the Horror Movie Segment in item-based algorithm.

Birds.² For this set of five movies, we then selected all combinations of three movies that had at least 50 users support, and chose 50 of those users randomly as our target segment and averaged the results.

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 the item-based algorithm. Our prediction shift results show that the segmented attack is

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>.

more effective against in-segment users than even the more knowledge-intensive average attack for the item-based collaborative algorithm. These results were also confirmed with a different segment based on a selection of Harrison Ford’s movies (*Star Wars*, *Return of the Jedi*, *Indiana Jones and the Last Crusade*, and *Raiders of the Lost Ark*), which for the sake of brevity we do not include. Figure 7 also shows that segmented attack is also successful against user-based algorithm.

5 Attack Profile Classification

In this section, we present our approach to attack detection and response based on profile classification: we analyze the characteristics of profiles and label each profile either as an authentic or an attack profile. We introduce two attributes that we have determined to be particularly effective at detecting segment attack, and show that a classifier based on these attributes can be extremely effective at detecting attack profiles that use the segment attack template. Prior work in detecting attacks in collaborative filtering systems have mainly focused on ad hoc algorithms for identifying basic attack models such as the random attack [6]. In contrast, we propose an alternate technique based on more traditional supervised learning. We show that a C4.5 classifier, using the proposed classification attributes, and built on a training set created through injecting system data with segment attack profiles, can be applied to unseen segment attack data with impressive results.

For this approach, a user’s profile is examined and based on characteristics of the profile, the entry is given a classification or Probability of Attack (PA) for the profile. Once each profile has been classified, this factor can be used to discount likely attack profiles to minimize their impact on other users. In Chirita et al. [6], a simple calculation was used to to discard likely attack profiles in user-based collaborative filtering by discounting the similarity with the PA through the following computation:

$$sim'_{u,v} = sim_{u,v} * (1 - PA_v) \quad (11)$$

where $sim_{u,v}$ is the Pearson correlation for profile u and neighbor profile v as calculated in equation (3), and PA_v is the PA for profile v . We apply a similar process to discount the attack profiles. For item-based algorithm, however, the discounting is slightly more complicated since similarity is being tabulated over items rather than users. As a result, we must incorporate the PA into the adjusted cosine similarity from equation (5) to discount the weight applied to each specific user profile in calculating the similarity between the items. The resulting item similarity is computed as follows:

$$sim'_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) * (r_{u,j} - \bar{r}_u) * (1 - PA_u)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2 * (1 - PA_u)} * \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2 * (1 - PA_u)}} \quad (12)$$

where $r_{u,i}$ represents the rating of user u on item i , \bar{r}_u is the average of the user u 's ratings, and PA_u is the PA score for profile u as before.

5.1 Generic Attributes for Detection

The hypothesis behind using generic attributes is based on the expectation that the overall signature of attack profiles differs from authentic profiles. This difference comes from two sources: the rating given the target item, and the distribution of ratings among the filler items. As many researchers in the area have theorized [3, 6, 4, 9], it is unlikely if not unrealistic for an attacker to have complete knowledge of the ratings in a real system. As a result generated profiles often deviate from rating patterns seen for authentic users. This variance may be manifested in many ways, including an abnormal deviation from the system average rating, or an unusual number of ratings in a profile. As a result, an attribute that captures these anomalies is likely to be informative in identifying reduced knowledge attack profiles (i.e., attack profiles with few filler items).

Prior work in attack profile classification has focused on detecting the general anomalies in attack profiles. Chirita et al. [6] introduced such a detection and response scheme. Their scheme focused heavily on an attribute they introduced for detecting the high rating deviations often associated with attack profiles. They referred to this attributes as *Rating Deviation from Mean Agreement* (RDMA), computed in the following way:

$$RDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i} - Avg_i|}{NR_i}}{N_u} \quad (13)$$

where N_u is the number of ratings in user u 's profile, $r_{u,i}$ is the rating given item i in user u 's profile, and NR_i is the number of overall ratings given to item i . In addition, their work hypothesised that attack profiles were likely to have a higher similarity with their top 25 closest neighbors than real users using Pearson correlation [12]. Their detection algorithm leveraged this by basing the mean agreement (Avg_i and NR_i) used in the RDMA on the subgroup of profiles whose average similarity over their top 25 neighbors was less than half of the maximum average top 25 similarity in the system. This value is then used to calculate the PA for profile u , by applying the following activation function:

$$PA_u = \begin{cases} 0, & \text{if } RDMA_u < RDMA_{Avg} \\ \frac{1}{e^{\alpha} - 1} (e^{\alpha \frac{RDMA_u - RDMA_{Avg}}{1 - RDMA_{Avg}}} - 1), & \text{otherwise} \end{cases} \quad (14)$$

These probabilities are then used to discount the weights of suspected attack profiles as described above. Their work showed this technique to be successful in detecting basic attacks with large filler sizes. As we will show, however, this algorithm has difficulty detecting the segment attack, and generally, low knowledge attacks attacks that have small filler sizes.

Instead, we propose using a variation of the RDMA measure which we call *Weighted Degree of Agreement* (WDA) that uses only the numerator of the

RDMA equation. This captures the sum of the differences of the profile’s ratings from the item’s average rating divided by the item’s rating frequency. The overall average \overline{WDA} is then subtracted from the profile’s value, WDA_u , as a normalizing factor.

5.2 Model-Specific Attributes for Detection

One weakness of generic attributes is that for smaller profile sizes, they have more difficulty in distinguishing an attack profile from an eccentric, but authentic, profile. To address the detection coverage gap at smaller filler sizes, we present an alternative approach for detection based on attack model similarity. In our approach, we detect attack profiles by comparing profile similarity to known attack models as done in traditional pattern classification. For this approach, known models could be used to build a training set where standard data mining techniques could then be applied to build a classifier. With a pattern matching approach, the closer an attacker mimics a known attack model, the greater the chance of detection. As a result, to avoid detection an attacker would theoretically have to deviate from the most effective models. By eliminating the benefit of the most effective profiles, the cost of an attack would increase since a larger attack would be required for a similar effect.

As shown in Section 2 attacks can be characterized based on the characteristics of their partitions I_{target} (the target item), I_S (selected items), and I_F (filler items). Model-specific attributes are those that aim to recognize the distinctive signature of a particular attack model. These attributes are based on partitioning each ratings profile in such a way as to maximize the profile’s similarity to a known attack model. Statistical features of the ratings that make up the partition can then be used as detection attributes. One useful property of partition-based features is that their derivation can be sensitive to additional information (such as time-series or critical mass data) that suggests likely attack targets.

Our detection model discovers partitions of each profile that maximizes its similarity to the attack model. To model this partitioning, each profile is split into two sets. The set P_{target} contains all items in the profile with the profile’s maximum ratings; the set P_{filler} consists of all other ratings in the profile. Thus the intention is for P_{target} to approximate $I_{target} \cup I_S$ and P_{filler} to approximate I_F . We will not attempt to differentiate I_{target} from I_S . We can then use statistical features of the partitions as detection attributes. For the segment attack model, the partitioning feature that maximizes the attack’s effectiveness is the difference in ratings of items in the $I_{target} \cup I_S$ compared to the items in I_F . Thus we introduce the *Filler Mean Target Difference* (FMTD) attribute. The attribute is calculated as follows:

$$FMTD_u = \left| \left(\frac{\sum_{i \in P_{target}} r_{u,i}}{|P_{target}|} \right) - \left(\frac{\sum_{k \in P_{filler}} r_{u,k}}{|P_{filler}|} \right) \right| \quad (15)$$

where $r_{u,i}$ is the rating given by user u to item i . The overall average \overline{FMTD} is then subtracted from $FMTD_u$ as a normalizing factor.

5.3 Detection Experimental Results

For our detection experiments, we used the same Movie-Lens 100K dataset³ used in Section 4. To minimize over-training, the dataset was split into two equal-sized partitions. The first partition was made into a training set, while the second was used for testing and was unseen during training. The training data was created by inserting a variety of segment attacks at various filler sizes that ranged from 3% to 100%. Once again to minimize over-training, all training data was created using the Harrison Ford segment while testing was executed on the 6 combinations of Horror segment movies. Specifically the training data was created by inserting a 1% segment attack against the Harrison Ford segment at a particular filler size, and generating the detection attributes for the authentic and attack profiles. This process was repeated 6 more times with 1% segment attacks against the same segment at a different filler size, and generating the detection attributes. For all these subsequent attacks, the detection attributes of only the attack profiles were then added to the original detection attribute dataset. This approach combined with the average attribute normalizing factor described above, allowed a larger attack training set to be created while minimizing over training for larger attack sizes that would affect the WDA distribution.

Based on the training data described above, C4.5 was used to build a binary profile classifier with $PA_u = 0$ if classified as *authentic* and $PA_u = 1$ if classified as *attack*. The resulting tree was used for all subsequent detection runs labeled *Segment Model Detection* and is depicted in Figure 9. The Chirita et al. algorithm was also implemented for comparison purposes (with $\alpha = 10$), and run on the test set described above and labeled *Chirita et al. Detection* [6]. To measure the effectiveness of the classification algorithms we used attack *precision* and *recall*. Attack precision is the ratio of true positives to all (true and false) positives, while attack recall is defined as the ratio of true positives to the sum of true positives and false negative. All segment attack results reflect the average over the 6 combinations of Horror segment movies described in Section 4. All prediction shift numbers for both user-based and item-based recommendation algorithms use the same parameters as in Section 4.

In our results, we present the detection capabilities of the proposed attributes for increasing the robustness of both user-based and item-based collaborative filtering against segment push attacks. Table 1 depicts the average confusion matrix for each detection algorithm averaged over filler sizes of 3%, 5%, 10%, 20%, 40%, 60%, 80%, and 100% for a segment attack of size 1% (i.e., containing attack profiles numbering to 1% of the original profile database).

As the confusion matrices show, both algorithms perform fairly well on average over the range of filler sizes although our Attack Similarity Model’s classification resulted in fewer false positives. A more in depth look at the performance

³ <http://www.cs.umn.edu/research/GroupLens/data/>

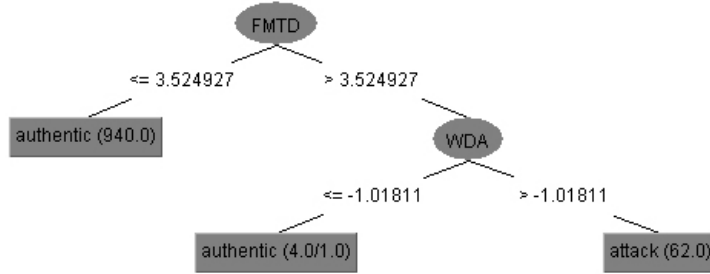


Fig. 9. C4.5 Decision tree using the proposed FMTD and WDA attributes.

Segment Model Detection			Chirita et al. Detection		
Authentic	Attack		Authentic	Attack	
942.3	0.7	Authentic	794.1	148.9	Authentic
0.0	9.0	Attack	2.7	6.3	Attack

Table 1. Comparison of confusion matrices for the Chirita et al. and the Segment Model algorithms, using 1% Horror segment attack averaged across filler sizes

of the Chirita et al. algorithm reveals the biggest gap in attack profile recall is at lower filler sizes while it performs very well at higher filler sizes. For the Attack Similarity Model, the precision improved from 81% at 3% filler size to 100% for filler sizes 40% or greater with the recall remaining at 100% throughout the range. The precision and recall results comparing these algorithms are depicted in Figure 10.

These detection algorithms were then used with user-based and item-based collaborative filtering as described in Section 5. As Figure 11 shows, although the Chirita et al. detection performs very well at higher filler sizes it has difficulty at lower filler sizes for even small attack sizes. The results also show the effect of training in the Segment Similarity Detection Model solely on small attack sizes. It should be noted that when the training data included higher attack sizes the overall classification performance at high attack sizes greatly improved, but at the cost of lower recall at small attack sizes. Based on these results, the low attack size biased training set was kept as the smaller attack sizes are likely to be more realistic. For item-based both detection methods significantly increased robustness resulting in insignificant prediction shift regardless of filler size up to an attack size of 15%.

To summarize, the Chirita et al. algorithm performs well in detecting basic attack models (such as the random attack) involving profiles with large filler sizes. However, it falls short when faced with more sophisticated attack profiles such as those based on the segment attack. Our classification-based approach

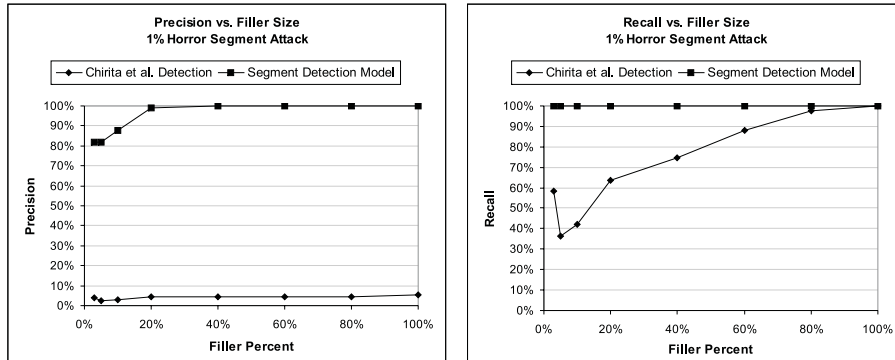


Fig. 10. Detection performance vs. filler size for 1% Horror segment attack

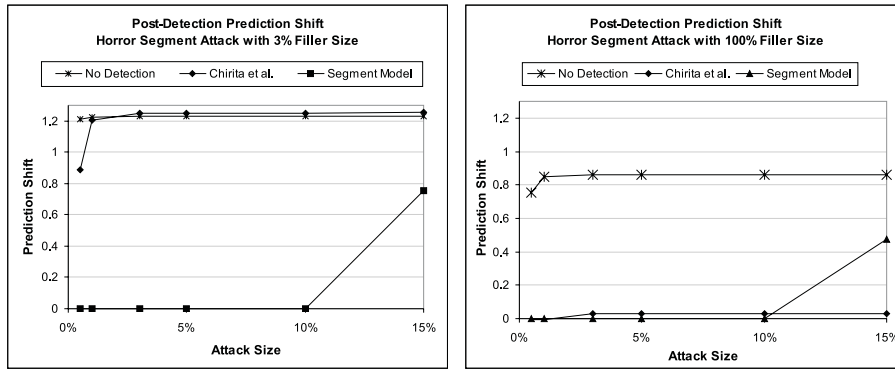


Fig. 11. User-based in-segment prediction shift for Horror segment attack at 100% filler size

using the proposed model-specific classification attribute is far more successful in detecting such attack profiles.

6 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. We have been able to model and empirically demonstrate successful attacks against the most common collaborative algorithms. We also have shown that although segment attack is effective against both user-based and item-based collaborative filtering, attack profile detection can be an effective way of increasing robustness for this type of attack.

This paper has shown several key findings in the area of attacks against recommender systems. We have shown that it is possible to mount successful attacks against such systems without a substantial knowledge of the system or users. The examination of segment attack, a very effective reduced knowledge attack, also demonstrated the vulnerability of the more robust item-based algorithm. In addition we proposed a profile classification and discounting algorithm which we call *Attack Similarity Model*, that can effectively identify attack profiles and provide a significant increase in robustness of collaborative filtering. Detection enhanced recommender systems, which combine collaborative recommendation with profile detection and discounting, seem likely to provide significant defensive advantages for recommender systems.

In our future work, we plan to deepen and extend our study of profile injection attacks. We will further develop ideas on detection and response and to evaluate their effectiveness against simulated and (given the availability of appropriate data) real-world attacks. In particular, we will develop detection algorithms that can automatically identify injected attack profiles based on many of the attack models described in this work. This will likely include additional generic and model-specific detection attributes to identify more general attack models. We are also exploring combining model-based attributes with time-series analysis to further improve robustness.

Our evaluation methodology relies on attack profiles that are generated in a fairly straightforward way from our attack models. We know that these attacks are effective, so they represent vulnerabilities that should be addressed. We plan to investigate what happens when an attack deviates from the attack model. Is the decrease in attack efficiency ameliorated by a decreased chance of detection from our model-based classifier? If we can ensure that the efficiency of attacks falls off faster than our ability to detect, then the model-based approach will represent an effective all-around defense.

References

1. Burke, R., Mobasher, B., Zabicki, R., Bhaumik, R.: Identifying attack models for secure recommendation. In: Beyond Personalization: A Workshop on the Next Generation of Recommender Systems, San Diego, California (2005)
2. Burke, R., Mobasher, B., Bhaumik, R.: Limited knowledge shilling attacks in collaborative filtering systems. In: Proceedings of the 3rd IJCAI Workshop in Intelligent Techniques for Personalization, Edinburgh, Scotland (2005)
3. Lam, S., Reidl, J.: Shilling recommender systems for fun and profit. In: Proceedings of the 13th International WWW Conference, New York (2004)
4. O'Mahony, M., Hurley, N., Kushmerick, N., Silvestre, G.: Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology* **4**(4) (2004) 344–377
5. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International World Wide Web Conference, Hong Kong (2001)
6. Chirita, P.A., Nejdl, W., Zamfir, C.: Preventing shilling attacks in online recommender systems. In: *WIDM '05: Proceedings of the 7th annual ACM international*

- workshop on Web information and data management, New York, NY, USA, ACM Press (2005) 67–74
7. Su, X.F., Zeng, H.J., Chen, Z.: Finding group shilling in recommendation system. In: Proceedings of the 14th international World Wide Web Conference (WWW05). (2005)
 8. O Mahony, M., Hurley, N., Silvestre, G.: Utility-based neighbourhood formation for efficient and robust collaborative filtering. In: Proceedings of the 5th ACM Conference on Electronic Commerce (EC 04). (2004) 260–261
 9. Mobasher, B., Burke, R., Bhaumik, R., Williams, C.: Effective attack models for shilling item-based collaborative filtering systems. In: Proceedings of the 2005 WebKDD Workshop, held in conjunction with ACM SIGKDD 2005, Chicago, Illinois (2005)
 10. Herlocker, J., Konstan, J., Borchers, A., Riedl, J.: 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 (1999)
 11. J.Herlocker, Konstan, J., Tervin, L.G., Riedl, J.: Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems* **22**(1) (2004) 5–53
 12. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work, ACM Press (1994) 175–186