

Defending recommender systems: detection of profile injection attacks

Chad A. Williams · Bamshad Mobasher · Robin Burke

Received: 4 May 2007 / Revised: 19 July 2007 / Accepted: 20 July 2007
© Springer-Verlag London Limited 2007

Abstract Collaborative recommender systems are known to be highly vulnerable to *profile injection attacks*, attacks that involve the insertion of biased profiles into the ratings database for the purpose of altering the system's recommendation behavior. Prior work has shown when profiles are reverse engineered to maximize influence; even a small number of malicious profiles can significantly bias the system. This paper describes a classification approach to the problem of detecting and responding to profile injection attacks. A number of attributes are identified that distinguish characteristics present in attack profiles in general, as well as an attribute generation approach for detecting profiles based on reverse engineered attack models. Three well-known classification algorithms are then used to demonstrate the combined benefit of these attributes and the impact the selection of classifier has with respect to improving the robustness of the recommender system. Our study demonstrates this technique significantly reduces the impact of the most powerful attack models previously studied, particularly when combined with a support vector machine classifier.

This research was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303 and the National Science Foundation IGERT program under Grant DGE-0549489.

C. A. Williams (✉)
Department of Computer Science, University of Illinois at Chicago,
Chicago, IL, USA
e-mail: cwilliam@cs.uic.edu

B. Mobasher · R. Burke
School of Computer Science,
Telecommunication, and Information Systems,
Center for Web Intelligence, DePaul University, Chicago, IL, USA
e-mail: mobasher@cs.depaul.edu

R. Burke
e-mail: rburke@cs.depaul.edu

Keywords Attack detection · Bias profile injection · Collaborative filtering · Recommender systems · Attack models · Support vector machines

1 Introduction

Recommender systems have become a staple of many e-commerce web sites, yet significant vulnerabilities exist in these systems when faced with what have been termed “shilling” attacks [5, 2, 9, 15]. We use the more descriptive phrase “profile injection attacks”, since promoting a particular product is only one way such an attack might be used. In a profile injection attack, an attacker interacts with a collaborative 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 these 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 [9] and [15].

Our prior work [2, 3] identified a number of attack models, based on different assumptions about attacker knowledge and intent. The overall conclusion is that an attacker wishing to “push” a particular product (make it more likely to be recommended) or “nuke” it (make it less likely to be recommended) can do so with a relatively modest number of injected profiles, with a minimum of system-specific

knowledge and with only the kind of general knowledge about likely user ratings distribution that one might find by reading the newspaper. We also know that profile injection attacks are not merely of theoretical interest, but have been uncovered at e-commerce sites.

As prior work has shown, if commercial recommendation systems are not protected, there is a very real risk the quality of the predictions and thus the consumer trust in the site can be compromised by attackers. The goal of this work is to address this vulnerability and provide tools and techniques web site owners may apply to protect their recommender services. Through techniques such as the one outlined in this paper, additional security and trust can be added to increase the robustness of recommendation systems used in the future for commercial sites.

The primary contribution of this paper is a description of an approach to detecting profile injection attacks with supervised classification. The technique is based on identifying characteristics of profiles that may be engineered to increase the influence of a malicious profile on the collaborative system. This is accomplished through a three pronged strategy to creating attributes to facilitate attack classification. This strategy combines attributes for detecting general ratings anomalies, similarity to reverse engineered attacks, and target concentrations; for use in a supervised approach to attack classification. A classifier is then built to distinguish attack profiles from genuine user profiles by constructing training data from authentic profiles and attacks generated by reverse engineered attack models. The combined effectiveness of this approach is then evaluated with the supervised classification algorithms k nearest neighbor (k NN), C4.5, and support vector machine (SVM). This study shows this defense technique when combined with the detection attributes described in this work and a robust classifier such as SVM, can nearly eliminate the impact of the most effective reverse engineered profile injection attacks for all but the largest attacks. We examine the impact the dimensions of attack type, attack intent, filler size, and attack size have on the effectiveness of such a detection scheme.

In Sect. 4, we provide a detailed description of our detection technique and the attributes used in this study. These attributes include both generic attributes that capture expected distribution of user data within profiles, as well as attributes based in the characteristics of well-known attack models. This is followed by our empirical analysis of the resulting detection classifier in Sect. 5.

2 Background and motivation

Researchers have shown that collaborative recommender systems, the most common type of web personalization system, are highly vulnerable to attack. Attackers can use automated

means to inject a large number of biased profiles into such a system, resulting in recommendations that favor or disfavor given items. Since collaborative recommender systems must be open to user input, it is difficult to design a system that cannot be so attacked. Researchers studying robust recommendation have therefore begun to study mechanisms for defending against such attacks.

Defense against profile injection can take many forms. Some collaborative algorithms are more robust than others against such attacks. Recent research has focused on techniques that can be used to protect the predictive integrity of collaborative recommenders from this type of malicious biasing. This work falls into two categories: techniques that increase the robustness of the recommender; and techniques for detecting and discounting biased profiles, like this work.

Motivating example

In this paper we consider attacks where the attacker's aim is to introduce a bias into a recommender system by injecting fake user ratings. In a profile injection attack, an attacker interacts with the recommender system to build within it a number of profiles with the aim of biasing the system's output. Such profiles will be associated with fictitious identities to disguise their true source.

An attack against a collaborative filtering recommender system consists of a set of attack profiles, each contained biased rating data associated with a fictitious user identity, and including 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 provide a hypothetical example to help illustrate the vulnerability of collaborative filtering algorithms, and will serve as a motivation for defending against such attacks. Consider, as an example, a recommender system that identifies books that users might like to read using a user-based collaborative algorithm [7]. 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 1 shows Alice's profile along with that of seven genuine users. An attacker, Eve, has inserted attack profiles (Attack1–3) into the system, all of which give high ratings to her book labeled Item6. Eve's attack profiles may closely match the profiles of one or more of the existing users (if Eve is able to obtain or predict such information), or they may be based on average or expected ratings of items across all users.

Suppose the system is using a simplified user-based collaborative filtering approach where the predicted ratings for Alice on Item6 will be obtained by finding the closest

Fig. 1 An example of a push attack favoring the target item Item6

	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		

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

On the other hand, if a system is using an item-based collaborative filtering approach, then the predicted rating for Item6 will be determined by comparing the rating vector for Item6 with those of the other items. Previous work has shown the item-based approach to be more robust, yet as this simple example demonstrates even more robust algorithms can still be vulnerable [11]. Obviously this example has been greatly simplified for illustrative purposes. While this paper uses user based collaborative filtering to illustrate the benefit of attack profile detection, as this latter observation illustrates detecting and eliminating such attack profiles could make other algorithms more robust as well. In real world systems both the product space and user database are much larger and more neighbors are used in prediction, but the same problem still exists.

Our overall aim is to protect collaborative recommenders from bias introduced by profile injection attacks. The intent of the approach examined in this work is to detect and respond to the most effective known attack models. Attackers wishing to evade detection will need to adopt less effective attacks, which by definition require greater numbers of profiles to produce the desired change in recommendation behavior. Larger attacks, however, are also conspicuous and in this way, we hope to render profile injection attacks relatively harmless.

3 Profile injection attacks

In this section, we present some of the dimensions across which profile injection attacks must be analyzed, and discuss the basic concepts and issues that motivate our analysis of detection in the rest of the paper. There are two main aspects of a profile injection attack that are needed to describe an attack on a collaborative recommender: the attack model, and the attack dimensions. Below we summarize how these two concepts relate to attack detection. See [2,3,5,12] for additional details.

3.1 Attack models

A profile-injection attack model is an approach for constructing a set of attack profiles, based on knowledge about the recommender system, its rating database, its products, and/or its users. The general form of these profiles is shown in Fig. 2. Each profile can be thought of as identifying four sets of items: a singleton target item i_t , a set of selected items with particular characteristics determined by the attacker I_S , a set of filler items usually chosen randomly I_F , and a set of unrated items I_\emptyset . Attack models can be defined by the methods by which they 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

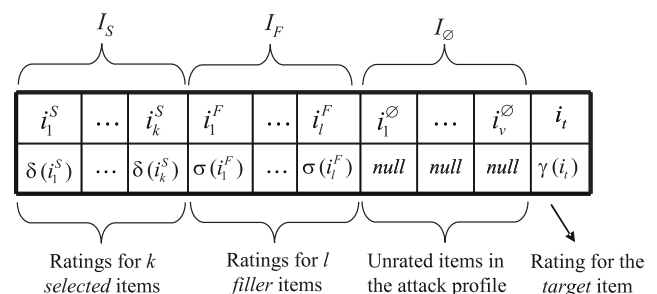


Fig. 2 The general form of a push/nuke attack profile

items and to the target item as defined by the functions δ , σ , and γ respectively. The set of selected items represents a small group of items that have been selected because of their association with the target item (or a targeted segment of users). For some attacks, this set is empty. On the other hand, the set of filler items represent a group of randomly selected items in the database which are assigned ratings within the attack profile. Since the selected item set is small, the size of each profile (total number of ratings) is determined mostly by the size of the filler item set. In our experimental results, we report filler size as a proportion of the size of I (i.e., the set of all items). The resulting 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 item being attacked, the target i_t , is r_{target} . Generally, in a push attack, $r_{\text{target}} = r_{\text{max}}$, while for a nuke attack, $r_{\text{target}} = r_{\text{min}}$, where r_{max} and r_{min} are the maximum and minimum allowable rating values, respectively.

Two basic attack models, introduced originally in [9] are the *random* and *average* attacks. Both of these models involve the generation of attack profiles using randomly assigned ratings given to some filler items in the profile. In the random attack, the assigned ratings are based on the overall distribution of user ratings in the database. In our formalism, I_S is empty, the contents of I_F are selected randomly, and the function σ generates random ratings centered on the overall average rating in the database. The average attack is very similar, except that the random ratings for each filler item in I_F are centered on the individual mean for each item; thus requiring considerably more information about the distribution of ratings within the target system.

Of these attacks, the average attack is by far the more effective, but it may be impractical to mount, given the degree of system-specific knowledge of the ratings distribution that it requires. Further, as we show in [3], it is ineffectual and hence unlikely to be employed against an item-based formulation of collaborative recommendation. Our own experiments yielded three additional attack models: the bandwagon, segment and love/hate attacks described below. See [2, 3, 5] for additional details.

The *bandwagon attack* is similar to the random attack, but it uses a small amount of additional knowledge, namely the identification of a few of the most popular items in a particular domain: blockbuster movies, top-selling recordings, etc. This information is easy to obtain and not dependent on any specifics of the system under attack. The set I_S contains these popular items and they are given high ratings in the attack profiles. In our studies, the bandwagon attack works almost as well as the much more knowledge-intensive average attack.

The *segment attack* is designed specifically as an attack against the item-based algorithm. Item-based collaborative recommendation generates neighborhoods of similar items,

rather than neighborhoods of similar users. The goal of the attack therefore is to maximize the similarity between the target item and the *segment items* in I_S . The segment items are those well-liked by the market segment to which the target item i_t is aimed. The items in I_S are given high ratings to increase the similarity between them and the target item; the filler items are given low ratings, to decrease the similarity between these items and the target item. This attack proved to be highly effective against the item-based algorithm as expected, but it also works well against user-based collaborative recommendation.

3.2 Attack dimensions

Profile injection attacks can be categorized based on the knowledge required by the attacker to mount the attack, the intent of a particular attack, and the size of the attack. From the perspective of the attacker, the best attack against a system is one that yields the biggest impact for the least amount of effort. While the knowledge and effort required for an attack is an important aspect to consider, from a detection perspective, we are more interested in how these factors combine to define the dimensions of an attack. From this perspective we are primarily interested in the dimensions of:

- **Attack model:** The attack model, specifies the rating characteristics of the attack profile (as described above).
- **Attack intent:** The intent of an attack describes the intent of the attacker. Two simple intents are “push” and “nuke”. An attacker may insert profiles to make a product more likely (“push”) or less likely (“nuke”) to be recommended. Another possible aim of an attacker might be simple vandalism – to make the entire system function poorly. Our work here assumes a more focused economic motivation on the part of the attacker, namely that there is something to be gained by promoting or demoting a particular product. (Scenarios in which one product is promoted and others simultaneously attacked are outside the scope of this paper).
- **Profile size:** The number of ratings assigned in a given attack profile is the profile size. The addition of ratings is relatively lower in cost for the attacker compared to the creating of additional profiles. However, there is the additional factor of risk at work when profiles include ratings for a large percentage of the ratable items. Real users rarely rate more than a small fraction of the ratable items in a large recommendation space. No one can read every book that is published or view every movie. So, attack profiles with many, many ratings are easy to distinguish from those of genuine users and are a reasonably certain indicator of an attack.
- **Attack size:** The attack size is the number of profiles inserted related to an attack. We assume that a

sophisticated attacker will be able to automate the profile injection process. Therefore, the number of profiles is a crucial variable because it is possible to build on-line registration schemes requiring human intervention, and by this means, the site owner can impose a cost on the creation of new profiles.

In our investigation we examine how these dimensions affect detection of profile injection attacks.

4 Detection of attack profiles

One of the main strengths of collaborative recommender systems is the ability for users with unusual tastes to get meaningful suggestions by the system identifying users with similar peculiarities. This strength is also one of the challenges in securing recommender systems. Specifically the variability of opinion makes it difficult to say with certainty whether a particular profile is an attack profile or the preferences of an eccentric user. It is unrealistic to expect all profiles to be classified correctly. The goals for detection and response will therefore be: minimize the impact of an attack, reduce the likelihood of a successful attack, and minimize any negative impact resulting from the addition of the detection scheme.

The attacks that we outlined above, work well against collaborative algorithms because they were created by reverse engineering the algorithms to devise inputs with maximum impact. Attacks that deviate from these patterns will be less effective than those that conform to them. Our approach to attack detection will therefore focus on recognizing attacks based on these reverse engineered attack models. An ideal outcome would be one in which a system could be rendered secure by making attacks against it no longer cost effective, where cost is measured in the attacker's knowledge, effort, and time. As discussed in Sect. 6, other techniques have been studied for defending against profile injection attacks as well, but in this work we focus on a profile classification approach.

In this section, we explain some of the unique challenges associated with attack profile classification, as motivation for detection attributes. Describing how conceptually these challenges might affect the robustness of a classifier used in this context, and specifically why the SVM algorithm is likely a good fit for this type of application. Finally, we summarize the collection of detection attributes which have been consolidated from several papers and combined in the experiments below [4,6,13,22].

4.1 Attack profile classification

Since we have some knowledge of what types of attacks are successful, we can treat attack identification as a traditional pattern classification problem in which we seek to classify

profiles as matching known attack models. It may be that some genuine users will be classified as attackers, with consequences that we explore later.

In this paper, we concentrate on identifying suspicious profiles by their aggregate properties. This is a classification approach which extends some of the features introduced originally in [6]. Our approach also differs since rather than constructing an ad-hoc classifier, we use training data based on our attack models to build a classifier to separate attack profiles from genuine users. The classification attributes are created using two different types of analysis. The first type is created by looking at the profile as a whole and is thus generic and not specific to any attack model. The second type is attack-model based, and generates attributes related to detecting characteristics of a specific attack model and target concentrations across profiles. We investigate using three common and well-understood classifier learning methods: simple nearest-neighbor classification using k NN, decision-tree learning using C4.5, and SVM. Due to the challenges mentioned above, the robustness of these algorithms across all dimensions of attack becomes critical to the success of the detection scheme. As we demonstrate in our experiments, using a more robust learning method such as SVM can have a significant impact on reducing the vulnerability of the system.

4.2 Classifier model

Applying supervised learning methods for attack classification on ratings profiles presents some significant challenges. The exponential number of combinations of attack types, possible attack targets, and selection of segment and filler items makes it infeasible to enumerate a training set using the ratings profiles alone. As a result some techniques must be applied to generalize the idea of an authentic or attack profile beyond the raw ratings data. To accomplish this, detection attributes are used to capture statistical features of a profile that when combined with other detection attributes together describe the *signature* of the profile.

In order to train the classifier, a training set first needs to be created. This is done by taking a set of profiles from the profile database; these profiles are assumed to be from non-malicious users and are labeled *authentic*. Into this training data a mixture of attack types at various attack sizes and filler sizes is injected and labeled *attack*. The detection attributes are then generated for each rating profile, and only the detection attributes and label of profile is kept as part of the training set. Training the classifier then follows traditional supervised machine learning methods.

Another challenge that separates attack classification from traditional classification is the competitive nature of the problem. In traditional classification problems, there is always the challenge of trying to account for data conditions or noise in

the unseen data that were not present in the training data. For attack classification, this problem is compounded by the fact that there is an adversary, the attacker, who benefits from and thus can be assumed to actively look for ways to take advantage of these conditions. Thus in order for a classification scheme to be robust against attacks; it not only needs the detection attributes to be flexible enough to capture deviations, it also needs a classification algorithm that is robust to *malicious noise*.

To identify such a classification algorithm, it is worth considering conceptually how classifications or the classification model is made and its vulnerabilities. Conceptually a learning scheme that combined observations across the entire training set as a whole would likely be more robust than an approach based on a more localized approach from a coverage perspective. Thus we propose a SVM classifier is likely to be more robust than other models since its classifier essentially incorporates all training examples simultaneously in evaluating a given profile. The SVM algorithm has been studied widely, in part due to its theoretical basis and properties of its decision boundary. Specifically it mathematically finds the optimal decision hyperplane with the largest margin per attribute. What this means for the adversarial classification problem is that all attributes are considered and weighted such that they all can meaningfully influence the classification. Conceptually this has the nice feature that it would likely be more difficult for an attacker to disguise their entire signature and still have an effective attack. However it also seems likely that unseen eccentric profiles that are far away from the norm could also be easily classified incorrectly.

To validate this intuition, we empirically compare SVM with classifiers built using a more localized approach. Specifically we compare SVM with the opposite extreme k NN which is based on localization in the form of similarity, and an algorithm in the middle, C4.5 which uses a sequence of individual attribute values to drive more generalized localization for classification. Consider k NN, while it generally is not considered as accurate as more sophisticated techniques for generalized datasets, it has been found to be quite accurate in determining classes tied to user similarity. Given this it would seem k NN would be a natural fit for this type of classification, however consider the vulnerabilities of its classification approach. Specifically it suffers from having a fixed weight, some distance measure, that applies to all attributes. As a result an attacker could take advantage of this and distance his profile from other known attacks with minimal change to the effect of the attack as shown in the experiments below. The C4.5 algorithm while to a lesser extent likely suffers a similar weakness. If its decision tree is built without pruning, it will over fit the training data and not perform well on unseen data. However when pruned, the number of attributes considered in classification is often reduced significantly. As a result it seems possible for an attacker to construct

profiles in such a way as to manipulate the small subset of attributes considered while still maintaining an effective attack that conforms to an attack signature in all other ways. Thus we would expect SVM to be the most robust followed by C4.5 and k NN in terms of maliciously being able to beat the classifier. In our experiments below, we empirically show support for this intuition.

4.3 Detection attributes

As described above, our approach is classification learning based on attributes derived from each individual profile. These attributes come in two varieties: generic and attack type-specific. The generic attributes are basic descriptive statistics that attempt to capture some of the characteristics that will tend to make an attacker's profile look different from a genuine user. The attack type-specific attributes are implemented to detect profile characteristics specifically associated with a known attack type.

4.3.1 Generic attributes

We expect the overall statistical signature of attack profiles will differ significantly from that of 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 [6,9,11,15], 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 will 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 attack profiles.

For the detection classifier's data set we have used a number of generic attributes to capture these distribution differences, several of which we have extended from attributes originally proposed in [6]. These attributes are:

Rating Deviation from Mean Agreement (RDMA) [6], is intended to identify attackers through examining the profile's average deviation per item, weighted by the inverse of the number of ratings for that item. The attribute is calculated as follows:

$$RDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i} - \bar{r}_i|}{R_{U,i}}}{N_u}$$

where N_u is the number of items user u rated, $r_{u,i}$ is the rating given by user u to item i , \bar{r}_i is the average rating of item i , and let $R_{U,i}$ be the number of ratings provided for item i by all users.

Weighted Degree of Agreement (WDA), is introduced to capture the sum of the differences of the profile’s ratings from the item’s average rating divided by the item’s rating frequency. It is not weighted by the number of ratings by the user, thus only the numerator of the RDMA equation.

Weighted Deviation from Mean Agreement (WDMA), designed to help identify anomalies, places a high weight on rating deviations for sparse items. We have found it to provide the highest information gain of the attributes we have studied. It differs from RDMA only in that the number of ratings for an item is squared in the denominator inside the sum, thus reducing the weight associated with items rated by many users. The WDMA attribute can be computed in the following way:

$$WDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i} - \bar{r}_i|}{R_{U,i}^2}}{N_u}$$

where U is the universe of all users u ; let P_u be a profile for user u , consisting of a set of ratings $r_{u,i}$ for some items i in the universe of items to be rated; let N_u be the size of this profile in terms of the numbers of ratings; and let $R_{U,i}$ be the number of ratings provided for item i by all users, and \bar{r}_i be the average of these ratings.

Degree of Similarity with Top Neighbors (DegSim) [6], captures the average similarity of a profile’s k nearest neighbors. As researchers have hypothesized attack profiles are likely to have a higher similarity with their top 25 closest neighbors than real users [6, 19]. We also include a second slightly different attribute *DegSim'*, which discounts the average similarity if the neighbor shares fewer than d ratings in common. We have found this variant provides higher information gain at low filler sizes.

Length Variance (LengthVar) is introduced to capture how much the length of a given profile varies from the average length in the database. If there are a large number of possible items, it is unlikely that very large profiles come from real users, who would have to enter them all manually, as opposed to a soft-bot implementing a profile injection attack. As a result, this attribute is particularly effective at detecting attacks with large filler sizes.

4.3.2 Type-specific attributes

Prior work has shown that the generic attributes are insufficient for distinguishing a true attack profile from eccentric but authentic profiles [13]. This is especially true when the profiles are small, containing fewer filler items. Such attacks can still be successful in influencing recommendation results, so we seek to augment the generic attributes with some that are designed specifically to match the characteristics of the attack types discussed above.

As shown in Sect. 3 attacks can be characterized based on the way their partitions i_t (the target item), I_S (selected items), and I_F (filler items) are constructed. Type-specific attributes attempt to recognize the distinctive signature of a particular attack type. These attributes are based on partitioning each profile in such a way as to maximize the profile’s similarity to one generated by a known attack type. Statistical features of the ratings that make up the hypothesized partitions can then be used as detection attributes.

Our detection model discovers a partitioning of each profile that maximizes its similarity to a particular attack type. To model this partitioning, each profile is split into two sets. The set $P_{u,T}$ contains all items in the profile that are hypothesized as targets of the attack, and the set $P_{u,F}$ consists of all other ratings in the profile. Thus the intention is for $P_{u,T}$ to approximate $\{i_t\} \cup I_S$ and $P_{u,F}$ to approximate I_F . (We do not attempt to differentiate I_T from I_S .) It is these partitions, or more precisely, their statistical features that we focus on for creating type specific detection attributes. It is important to note that this type specific partitioning can be applied for either push and nuke attacks by selecting the hypothesized target set to favor either high rated items or low rated items respectively.

For detecting the distinctive signatures of attacks, there are a couple of measures we have found useful across several of the attack detection models. These attributes are designed to identify characteristics of the filler partition that may indicate the profile was not created by an authentic user. All of these attributes are calculated using the hypothesized filler partition for the profile identified by that specific attack detection model. These measures are:

Filler Mean Variance (FMV), the variance of the individual ratings in the hypothesized filler partition from the average rating for each of those items. The intuition behind this attribute is to capture abnormally high or low variances between the individual mean of each item and the ratings of the filler items of the profile in question. For example, since the filler items of average attack type by design closely follow the average rating on all items, one would expect the FMV to be below that of the average authentic profile. The FMV for a given profile can be calculated as the variance of the individual ratings in the hypothesized filler partition from the average rating for each of those items. The FMV for a given user u and attack detection model m , represented by $FMV_{u,m}$, can be calculated as

$$FMV_{u,m} = \sum_{i \in P_{u,F_m}} \frac{(r_{u,i} - \bar{r}_i)^2}{|P_{u,F_m}|}$$

where P_{u,F_m} is the partition of the profile of user u hypothesized to be the set of filler items F by model m , $r_{u,i}$ is the rating user u has given item i , \bar{r}_i is the mean rating of item i

across all users, and $|P_{u,F_m}|$ is the number of ratings in the hypothesized filler partition of profile P_u by model m .

Filler Mean Difference, which is the average of the absolute value of the difference between the user's rating and the mean rating for the hypothesized filler items (rather than the squared value as in the variance).

Filler Average Correlation, the correlation between the filler ratings in the profile and the average rating for each item. These derived attributes are used to identify a "best fit" partitioning of each profile under the assumption that the profile has been generated as part of an attack of a particular type.

Average attack model—The average attack type divides the profile into two partitions: the target item given an extreme rating, and the filler items given other ratings. The model essentially just needs to select an item to be the target and all other rated items become fillers. For this attack type, the partitioning is selected such that the ratings placed in the filler partition minimizes the FMV, since for average attack the filler ratings closely match average score for each item.

Random attack model—Like the average attack model, this model divides the ratings into the same partitions with the target partition being a single rating. The partitioning is determined by selecting the filler items such that the ratings placed in the filler partition minimize the Filler Average Correlation, since random ratings are unlikely to correlate with the real item means.

Group attack model—The partitioning of the group attack model is created in a different manner. All ratings in the profile that are given the profile's maximum rating are placed in the target partition, and all other ratings become the filler items. Using this same partitioning attributes can be created to detect both the bandwagon and segment attack types. For bandwagon attacks, analysis of the filler ratings is identical to the random attack type. For the segment attack type, the feature that maximizes the attack's effectiveness is the difference in ratings of items in the $P_{u,T}$ set compared to the items in $P_{u,F}$. Thus we introduce the *Filler Mean Target Difference* (FMTD) attribute, which is the difference between the mean of the ratings in the target partition and the mean of the ratings in the filler partition. The attribute is calculated as follows:

$$FMTD_u = \left| \left(\frac{\sum_{i \in P_{u,T}} r_{u,i}}{|P_{u,T}|} \right) - \left(\frac{\sum_{k \in P_{u,F}} r_{u,k}}{|P_{u,F}|} \right) \right|$$

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

Target Focus Model—All of the attributes thus far have concentrated on inter-profile statistics; target focus, however,

concentrates on intra-profile statistics. Here we are seeking to make use of the fact that a single profile cannot really influence the recommender system. Only a substantial attack containing a number of targeted profiles can achieve this result. It is therefore profitable to examine the density of target items across profiles. One of the advantages of the partitioning associated with the model-based attributes described above is that a set of suspected targets is identified for each profile. For the *Target Model Focus* attribute (TMF), we calculate the degree to which the partitioning of a given profile focuses on items common to other attack partitions, and therefore measures a consensus of suspicion regarding each profile. To calculate TMF for a profile, first we define F_i , the degree of focus on a given item, and then select from the profile's target set the item that has the highest focus and use its focus value.

4.4 Attack response and system robustness

Once attack profiles have been detected, the question then becomes how the system should respond in order to eliminate or reduce the bias introduced by the attack. Ideally all attack profiles would be ignored and the system would function as if no bias had been injected. However a more likely scenario is there are a number of profiles that are suspected of being part of an attack without 100% certainty. If such a suspicion could be quantified reliably, the probability that a profile was part of an attack could be used as a weight to discount the contribution of such questionable profiles toward any recommendation the system makes. In our experiments here, we use the simpler method of ignoring profiles labeled as attacks when making predictions.

Although we have focused primarily on the direct affect of the push and nuke attacks on the target items, it is worth mentioning that bias in the overall system is also an important aspect of robustness. For a system to be considered robust, it should not only be able to withstand a direct attack on an item with minimal prediction shift; it should also be able to provide just as accurate predictions for all other items.

5 Experiments

In our experiments we have used the publicly available Movie-Lens 100K dataset.¹ This dataset consists of 100,000 ratings on 1,682 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). Each user in the dataset has rated at least 20 movies.

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

5.1 Recommendation algorithm

We used the standard user-based collaborative recommendation algorithm using k -nearest-neighbor prediction [7, 20]. 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 k NN-based algorithm operates by selecting the k most similar users to the target user, and formulates a prediction by combining the preferences of these users. Similarity is measured using Pearson's r -correlation coefficient: similar users are those whose profiles are highly correlated with each other. In our implementation, we use a value of 20 for the neighborhood size, and we filter out all neighbors with a similarity of less than 0.1.

Once the most similar users are identified, we use the following formula to compute the prediction for an item i for target user u .

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

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 $\text{sim}_{u,v}$ is the mean-adjusted Pearson correlation described above. If the profiles correlate less than 3% of the items, the weight of the contribution to the prediction for that user is reduced to the number of correlated items over 3% of the items.

5.2 Evaluation metrics

There has been considerable research in the area of recommender systems evaluation [8]. Some of these concepts can also be applied to the evaluation of the security of recommender systems, but in evaluating security, the vulnerability of the recommender to attack is of more interest than the raw performance. To compare different classification algorithms, we are interested primarily in measures of classification performance. An accurate classifier will prevent attack profiles from having an impact. One additional factor that we identified in prior research is errors induced by false positives. Many of the algorithms classify real profiles as attackers, thereby potentially impacting the accuracy of the recommendations produced. It is therefore important to measure the impact of attack detection on recommendation accuracy.

For measuring classification performance, we use the standard binary classification measurements of specificity and sensitivity. The basic definition of *specificity* and *sensitivity*

can be written as:

$$\text{sensitivity} = \frac{\# \text{ true positives}}{\# \text{ true positives} + \# \text{ false negatives}}$$

$$\text{specificity} = \frac{\# \text{ true negatives}}{\# \text{ true negatives} + \# \text{ false positives}}$$

Since we are primarily interested in how well the algorithms detect attacks, we examine these metrics with respect to attack identification. Thus *# true positives* is the number of correctly classified attack profiles, *# false positives* is the number of authentic profiles misclassified as attack profiles, and *# false negatives* is the number of attack profiles misclassified as authentic profiles. Thus *sensitivity* measures the proportion of attack profiles correctly identified, and *specificity* measures the proportion of authentic profiles correctly identified.

In addition to these classification metrics, we are also interested in measuring the effect of discounting misclassified authentic profiles on predictive accuracy. We evaluate this impact by examining a commonly used metric for evaluating recommender predictive accuracy, *mean absolute error* (MAE). Assume that the set T is a set of ratings in a test set, then the MAE of a recommender system trained on an authentic rating set R can be calculated as follows:

$$\text{MAE} = \frac{\sum_{t \in T} |t_{u,i} - p_{u,i}|}{\|T\|}$$

where $t_{u,i}$ is a rating in T for user u and item i , $p_{u,i}$ is the predicted rating for user u and item i , and $\|T\|$ is the number of ratings in the set T .

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 [15] in measuring stability via prediction shift. Average prediction shift is defined as follows. Let U_T and I_T be the sets of users and items, respectively, in the test data. 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.

5.3 Experimental setup

The attack detection and response experiments were conducted using a separate training and test set by partitioning the ratings data in half. The first half was used to create training data for the attack detection classifiers used in later experiments. For each test the 2nd half of the data was injected with attack profiles and then run through the classifier that had been trained on the augmented first half of the data. This

approach was used since a typical cross-validation approach would be overly biased as the same movie being attacked would also be the movie being trained for. Thus requiring the assumption that the system had a priori knowledge of which item(s) would be attacked.

The training data was created by inserting a mix of the attack types described above for both push and nuke attacks at various filler sizes that ranged from 3 to 100%. The attacked movies in the training sets were chosen at random from movies that had between 80 and 100 ratings; about 1/4 of the movies in the database have more ratings. This range was selected so that there are enough ratings to balance the somewhat large training attack, while still making the training sensitive to smaller attacks on less frequently rated items. Specifically the training data was created by inserting the first attack at a particular filler size, and generating the detection attributes for the authentic and attack profiles. This process was repeated 18 more times for additional attack types and/or filler sizes, and generating the detection attributes separately. 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 due to the high percentage of attack profiles that make up the training set (10.5% total across the 19 training attacks).

The detection attributes were then automatically generated based on the augmented dataset and a class attribute (authentic/attack) was added. For these experiments we use 25 detection attributes:

- 6 generic attributes: WDMA, RDMA, WDA, Length Variance, DegSim ($k = 450$), and DegSim' ($k = 2, d = 963$);
- 6 average attack model attributes (3 for push, 3 for nuke): Filler Mean Variance, Filler Mean Difference, Profile Variance;
- 4 random attack model attributes (2 for push, 2 for nuke): Filler Mean Difference, Filler Average Correlation;
- 4 group attack model attributes for bandwagon attack (2 for push, 2 for nuke): Filler Mean Difference, Filler Average Correlation;
- 4 group attack model attributes for segment attack (2 for push, 2 for nuke): Filler Mean Target Difference, Filler Mean Variance and,
- 1 target detection model attribute: Target Model Focus.

5.4 Classifier performance results

Based on the training data and method described above, binary classifiers were built to classify profiles as either *attack* or *authentic*. For comparison 3 classifiers were implemented: k NN, C4.5, and SVM. To classify unseen profiles with k NN,

the detection attributes of the profiles are used to find the 9 nearest neighbors in the training set to determine the class using Pearson correlation for similarity. The C4.5 and SVM classifier's are built in a similar manner such that they classify profiles based on the detection attributes only. The C4.5 classifier uses reduced error pruning and a confidence factor of 0.25 [18]. For all experiments below, the attacks examined are push attacks. All classifiers and classification results were created using Weka [23].

In all classification experiments, to ensure the generality of the results, 50 movies were selected randomly that represented a wide range of average ratings and number of ratings. Each of these movies was attacked individually and the average is reported for all experiments. The results reported below represent averages across all profiles in the test set and test movies.

In the first set of experiments we examine how the 3 classifiers compare at detecting the *average attack*, one of the more difficult to detect attack models. Figures 3 and 4 compare the classification performance of each of the classifiers to a 1% average attack across various filler sizes. As Sect. 5.2 explains, in this detection context sensitivity is the percent of attack profiles correctly identified; and specificity is the percent of authentic profiles correctly identified. As the sensitivity results show both SVM and C4.5 are nearly perfect at identifying all the attack profiles correctly, while the k NN classifier has some difficulty at low filler sizes. However, looking at the specificity we see the opposite is true with C4.5 and SVM misclassifying far more authentic profiles

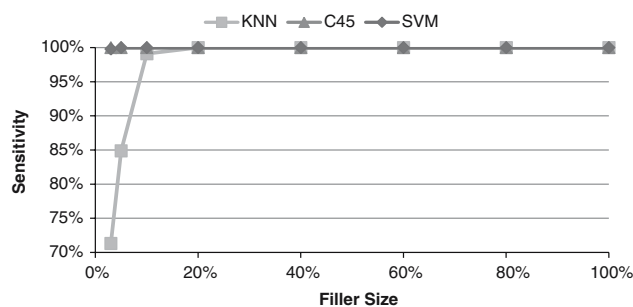


Fig. 3 Sensitivity comparison vs. filler size for 1% average attacks

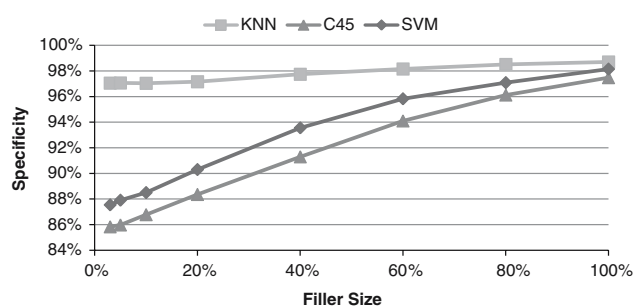


Fig. 4 Specificity comparison vs. filler size for 1% average attacks

than k NN; although this gap diminishes at higher filler sizes. This is not particularly surprising since there is often a trade off associated with sensitivity and specificity. Still, SVM has the best combination of sensitivity and specificity across the entire range of filler sizes for a 1% attack.

When analyzing the classifier accuracy, both type I (false positive—specificity) and type II (false negative—sensitivity) errors are important. Type I errors mean that real users are labeled as attackers; type II errors result in attackers slipping past our detection algorithm. However, as we show below, false positives are not particularly harmful if the system has a sufficiently large user base. This means that recall (finding all of the attackers) should be valued more than precision (detecting only real attackers).

5.5 Recommender impact analysis

Two questions follow from these results:

Accuracy: As Fig. 4 shows, all three detection algorithms incorrectly classify a portion of the authentic users as attack users. Does the system still make good predictions even when some genuine users are labeled as attackers and therefore ignored?

Robustness: As Fig. 3 shows, all three algorithms also allow some attackers to slip past undetected, but the vast majority of attack profiles are correctly identified. To what extent does this detection ability succeed in defending the system against the influence of an attack?

To answer these questions, we experimented with a version of the user-based recommendation algorithm in which users identified as attackers were ignored in the generation of recommendations. For examining the question of accuracy, we look at the Mean Absolute Error (MAE) for the system's predictions. To compute this value, we compare predicted and actual ratings over all users and movies in the original test set, applying the classifier such that all authentic users labeled as attackers are not included in predictions. If the detection system discarded too many real profiles (false positives), we would expect that prediction accuracy would go down and the error would go up. Figure 5 shows that, although C4.5, SVM, and k NN detection incorrectly classified some authentic users, the algorithms still are quite accurate with less than 0.02 on a rating scale of 1–5 or less than 1% difference from the system without detection. In fact with a 90% confidence interval, the differences between both k NN and SVM when compared to no detection is not statistically significant as Fig. 5 shows.

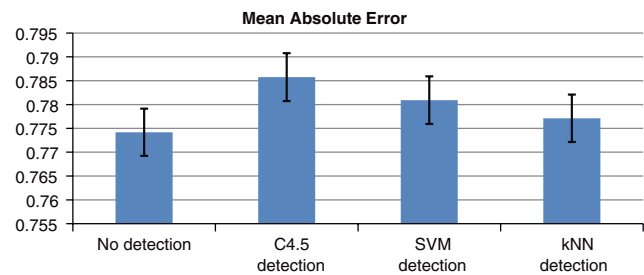


Fig. 5 Mean absolute error by detection algorithm shown with a 90% confidence interval

The question of robustness can be addressed in several ways, below we examine it with respect to *Prediction Shift*, the extent to which the system's predicted rating for the target item changes as a result of the attack. For the prediction shift experiments, attack classification was incorporated by eliminating any user from similarity consideration if it was classified as an attack user. User-based k NN collaborative recommendation was then applied with a neighborhood size of $k = 20$.

Figure 6 shows the resulting prediction shift caused by an average attack across all filler sizes and attack sizes from 0.5% to 15%. As the figure shows without detection, the system's predictions can be shifted significantly for even small attack sizes. However with detection, all three algorithms significantly reduce the range of attacks that are successful, particularly at low attack sizes. The more interesting aspect of these results are the differences in robustness of the 3 algorithms built on the same attributes and training set at different points in this filler size and attack size range. As Figs. 6 and 7 depict, while k NN may have superior specificity, its reduced sensitivity at small filler sizes becomes readily apparent. The reason for this difference is due to k NN's reliance on a good similarity metric for meaningful predictions, in this case the Pearson correlation coefficient. While this correlation coefficient generally performs well for ratings data; when there are few corated items, as would be the case for low filler sizes, it is prone to error due to the reduced overlap upon which it bases the correlation. The C4.5 and SVM algorithms, on the other hand, rely on matching profile characteristics to the decision space defined by the entire training set and are thus more robust to small filler sizes than k NN for this problem. By comparing the C4.5 and SVM classifiers (Fig. 7) each has an area which they dominate. The C4.5 algorithm performs slightly better at filler sizes of 10% or less when the attack size is 10% or more. The SVM algorithm, however, dominates for attack sizes less than 10%, allowing no resulting prediction shift over that entire range.

It is important to note that while the detection algorithm directly impacts the number of attack profiles used by the prediction algorithm, this does not necessarily mean the area where the most profiles slip through will result in the largest

Fig. 6 Prediction shift for average attack across the dimensions of filler size and attack size: **a** no detection and **b** k NN detection

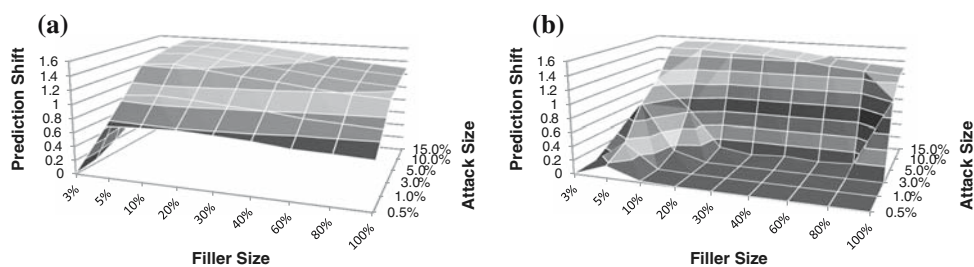


Fig. 7 Prediction shift for average attack across the dimensions of filler size and attack size: **a** C4.5 detection and **b** SVM detection

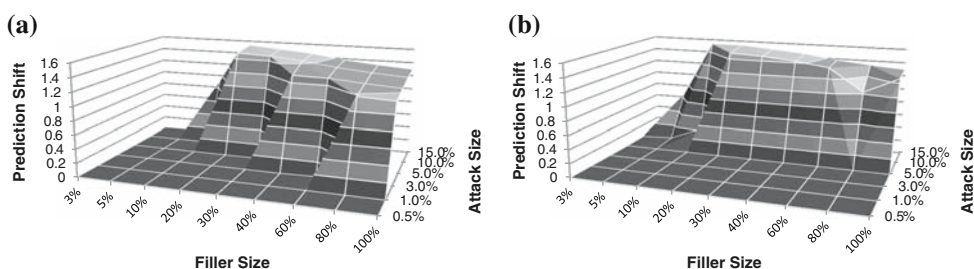


Fig. 8 Prediction shift for random attack across the dimensions of filler size and attack size: **a** no detection and **b** k NN detection

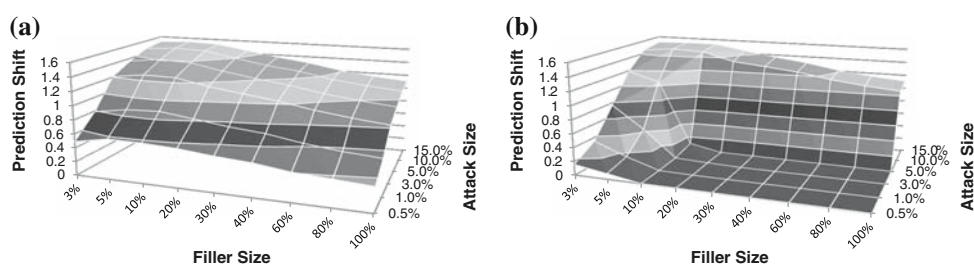
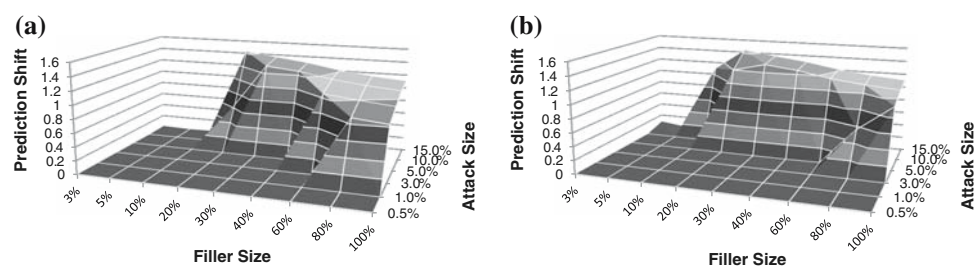


Fig. 9 Prediction shift for random attack across the dimensions of filler size and attack size: **a** C4.5 detection and **b** SVM detection



prediction shift. This phenomenon can be seen in Fig. 6b where the greatest prediction shift for a 1% attack with k NN detection occurred with a 5% filler size, even through the filler size with the lowest sensitivity (Fig. 3) was at 3%. This is due to the 5% attack profiles being far more effective per profile than the 3% ones as Fig. 6a shows. Thus, the most effective attack is one that both avoids detection and imparts the greatest impact to the recommender. The prediction shift surfaces shown in this work intend to highlight the impact of the combination of these two on resulting recommender system.

Next in Figs. 8 and 9 we examine the effectiveness of each of these algorithms at protecting against the random attack. Similar to the results for average attack, all three classifiers reduce the impact of the attack but SVM and C4.5 prove more robust at small filler sizes. Also once again SVM does slightly better than C4.5 for lower attack sizes while C4.5 has the slight edge at low filler sizes and large attack sizes. While either could be argued as being preferable, the areas of the SVM detector's weakness, high attack sizes, could be more easily covered by also employing an anomaly detection technique [1].

6 Related Work

Research related to improving robustness has established that hybrid and model-based recommendation offer a strong defense against profile injection attacks, significantly reducing the impact of attacks for the most part [11, 14]. Other work by Zhang et al. [25] has shown singular value decomposition (SVD) techniques can also help reduce the effects of attacks. Massa and Avesani [10] introduced a trust network approach to limit the influence of biased users. O'Mahony et al. [17] developed several techniques to defend against the attacks described in [9, 15], including new strategies for neighborhood selection and similarity weight transformations.

However robust an algorithm may be, it is impossible to have complete security against profile injection attacks. A collaborative system is designed to adjust its behavior in response to user inputs, and in theory, an attacker could swamp the system with so many profiles as to control it completely. One common defense is to simply make assembling a profile more difficult. A system may require that users create an account and perhaps respond to a captcha² before doing so. This increases the cost of creating bogus accounts (although with offshore data entry outsourcing available at low rates, the cost may still not be too high for some attackers.) Such measures come at a high cost for the system owner as well, however – they drive users away from participating in collaborative systems, systems which rely on user input to function. In addition, such measures are totally ineffective for recommender systems based on implicit measures such as usage data mined from web logs.

Other research efforts have been aimed at detecting and preventing the effects of profile injection attacks. Chirita et al. [6] proposed several metrics for analyzing rating patterns of malicious users and evaluate their potential for detecting such attacks. Su, et al. [21] developed a spreading similarity algorithm in order to detect groups of similar attackers. In Burke et al. [4] a model-based approach to detection attribute generation was introduced and shown to be effective at detecting and reducing the effects of random and average attack types. A second model-based approach for detecting attacks that target groups of items was introduced in Mobasher et al. [13] and shown to effectively detect the segment attack. Other work has examined a more unsupervised approach based on anomaly detection to identify attacks. Bhaumik et al. [1] demonstrated that X-bar and confidence interval control limit anomaly detection techniques could be used effectively to identify items and time periods an item was under attack for even small attack sizes. Zhang et al. [24] introduced a heuristic based approach to adapting time-series windows to more accurately detect attack events based on changes in averages and entropy between periods.

² www.captcha.net/.

O'Mahony et al. [16] examined the problem of deviant ratings at a more general level trying to detect and eliminate any ratings that degraded the quality of predictions whether malicious or natural. Their work showed that such a technique could be used to increase robustness with minimal impact to accuracy or coverage.

7 Conclusion

Profile injection attacks have been shown to be effective threats to the robustness of collaborative recommender systems. Our work and others have pointed out the vulnerabilities shared by the most commonly-implemented collaborative algorithms. In this paper, we demonstrate a supervised classification learning approach can add significant robustness to profile injection attacks. Furthermore our results demonstrate the selection of classifier algorithm is also an important factor in maximizing the protection this type of scheme can offer. Specifically a classification algorithm should be chosen that is not easily beat by a malicious user manipulating individual features of the attack profile. As this work shows when combined with a robust classification algorithm such as SVM, significant robustness can be obtained against all but the largest attack sizes while having insignificant impact to predictive accuracy.

Several outstanding questions remain, however. We have incorporated attack-specific feature extraction into the classifiers. Some preliminary work on detecting attacks that deviate from these reverse engineered models was done with some success using a *k*NN based detection technique [22]. Some preliminary results, not included here for reasons of space, indicate a more robust algorithm like SVM may provide additional protection against these types of attacks as well. Other preliminary work also indicates that the classifiers trained on average and random attacks do work well on the other attack models that we have identified. Further research is necessary to determine how the choice of classification algorithm may affect robustness of our detection method for nuke attacks. Another area to explore would be the robustness a hybrid of profile classification and anomaly detection techniques could provide. In general, it remains to be shown whether a theoretical approach can be used to prove the robustness of any non-trivial defense mechanism to the realm of any possible attack.

Our model of detection described above incorporates multiple dimensions, such as time series and critical mass information. The results reported here, however, do not incorporate temporal properties and use the profiles in isolation without attempting to identify common items under attack. We expect that taking these features into account will provide further enhancements to our detection accuracy.

References

1. Bhaumik R, Williams C, Mobasher B, Burke R (2006) Securing collaborative filtering against malicious attacks through anomaly detection. In: Proceedings of the 4th workshop on intelligent techniques for web personalization (ITWP'06), Held at AAAI 2006, Boston, July 2006
2. Burke R, Mobasher B, Bhaumik R (2005) Limited knowledge shilling attacks in collaborative filtering systems. In: Proceedings of the 3rd IJCAI workshop in intelligent techniques for personalization, Edinburgh, Scotland, August 2005
3. Burke R, Mobasher B, Williams C, Bhaumik R (2005) Segment-based injection attacks against collaborative filtering recommender systems. In: Proceedings of the international conference on data mining (ICDM 2005), Houston, December 2005
4. Burke R, Mobasher B, Williams C, Bhaumik R (2006) Detecting profile injection attacks in collaborative recommender systems. In: Proceedings of the IEEE joint conference on e-commerce technology and enterprise computing, e-commerce and e-services (CEC/EEE 2006), Palo Alto, CA, June 2006
5. Burke R, Mobasher B, Zabicki R, Bhaumik R (2005) Identifying attack models for secure recommendation. In: Beyond personalization: a workshop on the next generation of recommender systems, San Diego, California, January 2005
6. Chirita P, Nejd W, Zamfir C (2005) 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 pp 67–74
7. Herlocker J, Konstan J, Borchers A, Riedl J (1999) 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
8. Herlocker J, Konstan J, Tervin LG, Riedl J (2004) Evaluating collaborative filtering recommender systems. *ACM Trans Inform Syst* 22(1):5–53
9. Lam S, Reidl J (2004) Shilling recommender systems for fun and profit. In: Proceedings of the 13th international WWW conference, New York, May 2004
10. Massa P, Avesani P (2004) Trust-aware collaborative filtering for recommender systems. *Lecture Notes Comput Sci* 3290:492–508
11. Mobasher B, Burke R, Bhaumik R, Williams C (2005) 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, IL, August 2005
12. Mobasher B, Burke R, Bhaumik R, Williams C (2007) Towards trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans Int Technol* (in press)
13. Mobasher B, Burke R, Williams C, Bhaumik R (2006) Analysis and detection of segment-focused attacks against collaborative recommendation. In: Lecture notes in computer science: Proceedings of the 2005 WebKDD workshop. Springer, Heidelberg
14. Mobasher B, Burke R, Sandvig JJ (2006) Model-based collaborative filtering as a defense against profile injection attacks. In: Proceedings of the 21st national conference on artificial intelligence (AAAI'06), Boston, MA, July 2006
15. O'Mahony M, Hurley N, Kushmerick N, Silvestre G (2004) Collaborative recommendation: a robustness analysis. *ACM Trans Internet Technol* 4(4):344–377
16. O'Mahony MP, Hurley NJ, Guérolé CM (2006) Silvestre. Detecting noise in recommender system databases. In: IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces, pp 109–115
17. O'Mahony MP, Hurley NJ, Silvestre G (2004) Utility-based neighbourhood formation for efficient and robust collaborative filtering. In: Proceedings of the 5th ACM conference on electronic commerce (EC04), pp 260–261, May 2004
18. Quinlan JR (1993) C4.5: Programs for machine learning. Morgan Kaufmann San Francisco (1994)
19. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) Grouplens: an open architecture for collaborative filtering of news. In: CSCW '94: Proceedings of the 1994 ACM conference on computer supported cooperative work, pp 175–186. ACM Press, New York
20. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international world Wide Web conference, Hong Kong, May 2001
21. Su X-F, Zeng H-J, Chen Z (2005) Finding group shilling in recommendation system. In: WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web, Chiba, Japan. ACM Press, New York pp 960–961
22. Williams C, Mobasher B, Burke R, Sandvig J, Bhaumik R (2006) Detection of obfuscated attacks in collaborative recommender systems. Held at the 17th European Conference on Artificial Intelligence (ECAI'06), Riva del Garda, Italy, August 2006
23. Witten IH, Frank E (2005) Data Mining: practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco
24. Zhang S, Chakrabarti A, Ford J, Makedon F (2006) Attack detection in time series for recommender systems. In: KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 809–814
25. Zhang S, Ouyang Y, Ford J, Makedon F (2006) Analysis of a low-dimensional linear model under recommendation attacks. In: SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA ACM Press, New York, pp 517–524